TECHNICKÁ UNIVERZITA V LIBERECI

Ústav zdravotnických studií

Studijní program: B 3944 Biomedicínská technika

Studijní obor: 3901R032 Biomedicínská technika

VÝVOJ APLIKACE PRO SBĚR A VYHODNOCENÍ DAT ZE SCINTILAČNÍHO POČÍTAČE

(Development of application for collecting and evaluation data from scintillation computer)

Jakub Mottl

Bakalářská práce

2011

TECHNICKÁ UNIVERZITA V LIBERCI
Ústav zdravotnických studií
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jakub MOTTL**
Osobní číslo:	**Z08000130**
Studijní program:	**B3944 Biomedicínská technika**
Studijní obor:	**Biomedicínská technika**
Název tématu:	**Vývoj aplikace pro sběr a vyhodnocení dat ze scintilačního počítače**
Zadávající katedra:	**Ústav zdravotnických studií**

Zásady pro vypracování:

Cíl výzkumu:
1) vytvořit fungující spojení mezi osobním počítačem a scintiačním počítačem po sériovém portu
2) zjistit možné způsoby ovládání a komunikace se scintilačním počítačem
3) vytvořit aplikaci vhodnou pro laboratorní využití scintilačním počítače
Východiska:
V Laboratoři neurofyziologie Fyziologického ústavu v Praze se nachází scintilační počítač (model Beckman LS 1804) z roku 1984. Tento přístroj disponuje pouze výstupem na tiskárnu a portem RS232.Data z výstupu scintilačního počítače (RS232) byla měla být ukládána do PC. Dalším krokem by bylo vytvoření programu, který by umožnil s daty dále pracovat - to znamená je editovat, ukládat a vyhodnocovat.
Předpoklady:
Mělo by být možné aplikaci vytvořit.
Metoda: Kvantitativní.
Technika:
Analýza problému a zjištění požadavků pracovníků ústavu a možností scintilačního počítače.
Místo a čas výzkumu:
Průběžný vývoj aplikace vyžaduje několik návštěv ústavu proložené delší časovou periodou, kdy se bude vytvářet nová verze aplikace.
Vzorek: Pracovníci ústavu, kteří budou dotazování na požadavky aplikace.
Konzultant: Ing. Jindra Drábková, Ph.D.

Rozsah grafických prací:

Rozsah pracovní zprávy: **50-70 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] Delphi 4.0 help. Dostupný z instalačního CD-ROM Borland Delphi 4.0.

[2] VarianAsync 32 help. Dostupný z instalačního souboru VarianAsync 32.

[3] VYKOPAL, J. Sériové rozhraní v Delphi 1. – 5. díl [online]. Dostupný z WWW: ¡http://www.builder.cz¿.

[4] MAX 232 data sheet [online]. Dostupný z WWW: ¡http://www.maxim-ic.com ¿.

[5] RS-232 overview [online]. Dostupný z WWW: ¡http://www.hw-server.com/rs232¿.

[6] Vše o RS-232 [online]. Dostupný z WWW: ¡ http://rs232.hw.cz¿.

[7] Bylund DB, Toews ML: Radioligand binding methods: practical guide and tips. Am J Physiol Lung Cell Mol Physiol 265:L421-429.1993.

[8] Bylund DB, Deupree JD, Toews ML: Radioligand binding methods for membrane preparations and intact cells. In: Receptor Signal transduction protocols, Methods in molecular biology 259. Humana Press, New Jersey, 2004.

Vedoucí bakalářské práce: **doc. MUDr. Jaromír Mysliveček, Ph.D.**
Ústav zdravotnických studií

Datum zadání bakalářské práce: **30. dubna 2010**

Termín odevzdání bakalářské práce: **30. dubna 2011**

prof. Dr. Ing. Zdeněk Kůs
rektor

L.S.

Technická univerzita v Liberci -13-

doc. MUDr. Jaromír Mysliveček, Ph.D.
ředitel

V Liberci dne 30. listopadu 2010

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že souhlasím s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum: 23.4.2011

Podpis:

Poděkování:

Děkuji mému vedoucímu práce Doc. MUDr. Jaromírovi Myslivečkovi, Ph.D. za odborný dohled a poskytnutí cenných rad a informací k mé bakalářské práci. Dále bych chtěl poděkovat Ing. Jindře Drábkové, Ph.D, za to že byla mým konzultantem a poskytla mi cenné rady a vedení. Mé díky bych také rád směřoval pracovníkům neurofyziologické laboratoře 1. lékařské fakulty Univerzity Karlovy v Praze, především pak PharmDr. Vladimírovi Farárovi a Petře Svatošové. Poděkování patří také mé rodině za podporu a trpělivost po celou dobu mého studia.

Anotace

Bakalářská práce se dělí na teoretickou a praktickou část. Teoretická část se zabývá problematikou, scintilačních detektorů, detekce beta záření pomocí kapalných scintilátorů, strukturou, zapojení a okolnostmi komunikace po sériovém portu, vývojovým prostředím Lazarus a teorií vazebných studií.

Praktická část popisovala, jakým způsobem bylo vznikala aplikace Boreas, sloužící pro sběr a vyhodnocení dat ze scintilačního počítače a seznamuje s nejdůležitějšími částmi aplikace.

Klíčová slova: scintilační detektor, detekce beta záření, sériový port, IDE Lazarus, vazebné studie

Annotation

The bachelor´s work is divided into a theoretical and a practical part. The theoretical part deals with theory of scintillation detectors, detection of beta particles with liquid scintillators, structure, connection and communication over serial port, integrated development environment Lazarus and theory of binding studies.

The practical part describe, how the Boreas application, which is design for collecting and evaluation data from scintillation computer, was created and introduce the most important pats of application.

Keywords: scintillation detector, detection of beta particles, serial port, IDE Lazarus, binding studies.

Obsah

Seznam zkratek

ARM	Advanced RISC Machine	Jedna z architektur procesorů
ASCII	American Standard Code for Information Interchange	Americký standardní kód pro výměnu informací
CPM	Counts per minute	Záblesků za minutu
CSV	Comma-separated values	Hodnoty oddělené čárkami
CTS	Clear to Send	Modem tímto signálem oznamuje terminálu, že komunikační cesta je volná
DPM	Disintegrations per minute	Rozpadů za minutu
DSR	Data Set Ready	Modem tímto signálem oznamuje terminálu, že je připraven komunikovat
DTR	Data Terminal Ready	Terminál tímto signálem oznamuje modemu, že je připraven komunikovat
FPC	Free pascal compiler	Kompilátor jazyka Free Pascal
GUI	Graphical User Interference	Grafické uživatelské rozhraní
IDE	Integrated development environment	Integrované vývojové prostředí
LCL	Lazarus Component Library	Knihovna komponent pro vývojové prostředí Lazarus
OOP	Object Orientated Programing	Objektově orientované programování
OPA	One point assay	Jednobodová saturace
RAD	Rapid application development	Vývojové prostředí pro rychlý vývoj aplikací
RTS	Request to Send	Terminál tímto signálem oznamuje modemu, že komunikační cesta je volná
RxD	Receive Data	Přijímání dat
TxD	Transmit Data	Vysílání dat
UART	Universal Asynchronous Receiver and Transmitter	Univerzální asynchronií vysílač a přijímač
XML	Extensible Markup Language	Rozšiřitelný značkovací jazyk

I. Úvod

Tato bakalářské práce vznikla kvůli scintilačnímu počítači Beckman LS 1801, který se nachází v laboratoři na Fyziologickém ústavu 1. lékařské fakulty univerzity Karlovy. Tento přístroj disponuje pouze výstupem na tiskárnu a portem RS232.

Na tomto přístroji nyní probíhá část výzkumu, konkrétně vazebné studie specifické vazby na receptorech. Jednotlivé údaje se přepisují z vytištěných papírů do počítače, kde se následně zpracovávají.

Cílem bakalářské práce bylo dosáhnout toho, aby data z výstupu scintilačního počítače (RS232) byla ukládána do PC. Dalším krokem bylo vytvoření programu, který by umožnil s daty dále pracovat - to znamená je editovat, ukládat a vyhodnocovat. Na vyhodnocení dat, která vycházejí ze scintilačního počítače samozřejmě existují komerční programy. Ostatně i výrobce scintilačního počítače (Beckman Coulter) v dnešní době k novým přístrojům dodává i software na vyhodnocení. Protože parametry a spolehlivost měření přístroje je dostatečná, není nutnost investovat 1 000 000,- Kč do nového přístroje jen proto, aby získali software na vyhodnocování. Na druhou stranu, výstup na papír z tiskárny a pak přepisování dat je nepohodlné, a je častým zdrojem chyb. Z toho důvodu bylo třeba vytvořit aplikaci, která by umožnila zachycení dat a jejich následné zpracování.

Aplikace, která byla v rámci této práce vytvořena, nese název Boreas.

II. Cíle práce

Hlavní cíl: Vytvořit funkční aplikaci pro sběr a vyhodnocení dat ze scintilačního počítače Beckman LS 1801

Dílčí cíle:

1. Vytvořit fungující spojení mezi osobním počítačem a scintilačním počítačem po sériovém portu
2. Zjistit možné způsoby ovládání a komunikace se scintilačním počítačem
3. Vytvořit aplikaci vhodnou pro laboratorní využití scintilačním počítačem

Teoretická část práce je rozdělena do čtyř kapitol. První kapitola bakalářské práce se zabývá scintilačním počítačem jako takovým, jeho součástí a fyzikálním principem scintilace. Druhá kapitola se věnuje problematice sériového portu, jeho vlastností a problémem komunikace po sériovém portu. Ve třetí kapitole jsou shrnuty informace o vývojovém prostředí Lazarus, ve kterém je aplikace Boreas vytvořena. Čtvrtá kapitola je věnována problematice vazebných studií, jejichž data aplikace zpracovává.

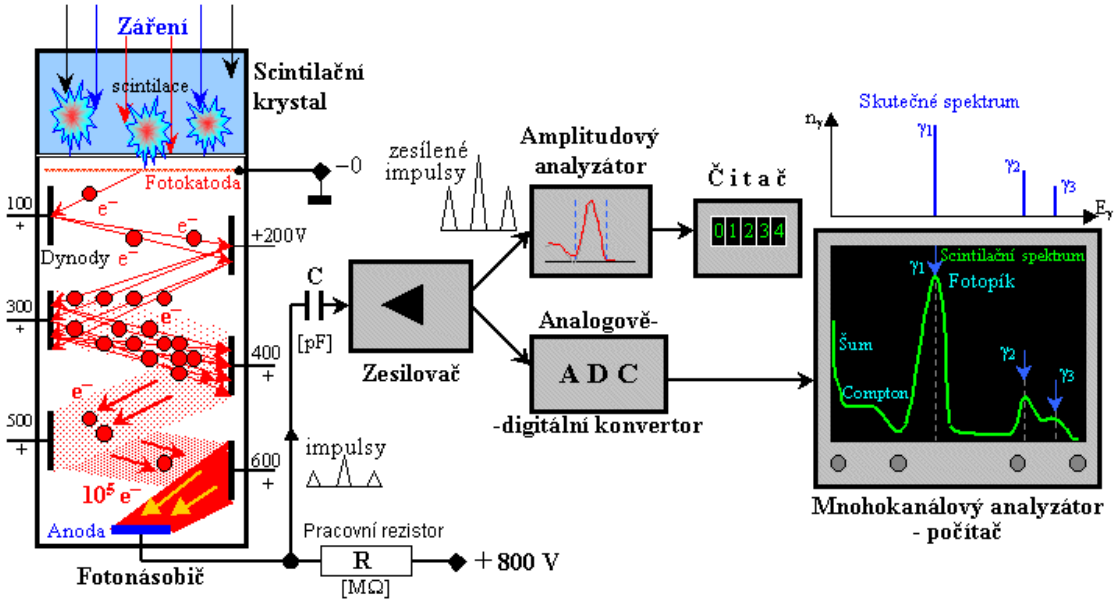
III. Teoretická část

1 Scintilační počítač

Scintilační počítač je poměrně komplikované zařízení, které v principu slouží k měření aktivity (radioaktivity) zdroje.

Scintilačních počítačů může existovat velké množství druhů a typů, u kterých záleží především na typu vzorků, které chceme měřit, jejich množství, skupenství, aktivitě a tak dále.

Základní součástí každého scintilačního počítače je scintilační detektor, který se skládá ze scintilátoru a fotonásobiče. (viz Obr. 1-1)



Obr. 1-1 Horní větev schématu představuje princip scintilačního detektoru a dolní větev schéma principu spektrometru.
V pravé části je na obrazovce typický tvar scintilačního spektra záření γ. Nad ní je skutečné čárové spektrum. [2]

Scintilační detektory jsou založeny na vlastnosti některých látek, které se používají v detektoru jako scintilátory, reagovat na ionizující záření světelnými záblesky (scintilacemi). Tyto záblesky jsou velmi slabé a musí být zesíleny pomocí fotonásobiče, aby bylo možné jejich energii převést na elektrický impulz jehož, amplitudu jsme na

výstupu detektoru schopni měřit.

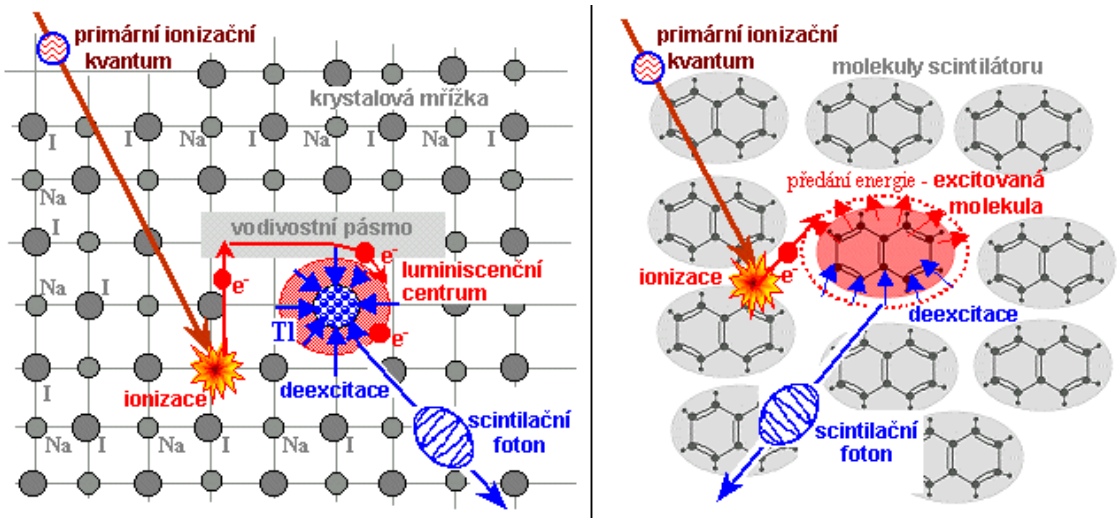
1.1 Scintilátor

Pro různé druhy záření a pro různě vysoké energie je nutno použít různé typy scintilátorů. Většina scintilátorů je anorganického, nebo organického původů, přičemž anorganické scintilátory jsou zejména krystaly některých látek, do nichž bylo dodáno několik atomů jiné látky. Takovýmto atomům se říká aktivátory.

Nejstarším používaným scintilátorem je sirník zinečnatý aktivovaný stříbrem ZnS(Ag), který byl použit třeba na stínítka skiaskopických rentgenových přístrojů. V minulosti se používal i kyanid platino-barnatý. Pro účely detekce záření γ se však nejčastěji používá jodid sodný aktivovaný thaliem - NaI(Tl), ve formě monokrystalu, kdy ionty thalia jsou zastoupeny v množství 1-2% z celkového množství atomů.

Princip fungování anorganického scintilátoru se pokusíme vysvětlit na příkladu krystalu NaI(Tl), který se používá na detekci záření γ. Pro tento scintilátor je jako u většiny anorganický scintilátorů důležitá krystalová mřížka, která je narušená atomy aktivátoru (zde thalia).

V krystalu potom v luminiscenčním centru dojde k zachycení a pohlcení fotonu záření γ atomem krystalu. Tento atom se dostane do excitovaného stavu a přesune se do vyšší energetické hladiny. Tento stav je ale velmi nestabilní a brzy dojde k návratu tohoto atomu do původní energetické hladiny. Přebytek energie je potom vyzářen ve formě sekundárního fotonu. Aktivátor (atom jiné látky dodaný do krystalu) potom v krystalu zvyšuje množství luminiscenčních center, což jsou místa v krystalu, kde může dojít k zachycení primárního fotonu ionizačního záření. Důležitým prvkem v anorganických scintilátorech je krystalická mřížka, jejíž struktura umožňuje vznik scintilačního efektu (viz Obr. 1.1-1 vlevo).



Obr. 1.1-1 Schématické znázornění vzniku scintilačního záření v anorganickém scintilátoru (vlevo) a v organickém scintilátoru. (vpravo) [2]

Výsledná scintilace je tvořena většinou několika stovkami sekundárních fotonů, v závislosti na absorbované energii primárního detekovaného kvanta.

Problémem NaI(Tl) krystalů je jejich velká křehkost a náchylnost na prudké změny teplot. Dalšími problémy je jejich vysoká hygroskopie a hydrolýza vzdušnou vlhkostí. Díky tomu se velice zhoršuje jejich detekční účinnost a proto je krystal většinou uložen ve světlotěsném, hermeticky uzavřeném hliníkovém pouzdře (málo stíní ionizační záření). Vnitřní strany pouzdra jsou opatřeny bílou reflexní vrstvou, která odráží světelné fotony na fotokatodu fotonásobiče. Pro obecnou detekci a spektrometrii záření γ se používají planární scintilační krystaly válcového tvaru o průměru kolem 2-7cm a výšky cca 2-8cm. Pro detekci měkkého γ a X záření pak tenké krystaly tloušťky 1-5mm s tenkým aluminiovým nebo beryliovým vstupním okénkem.

Organické scintilátory se chovají trochu jinak. Na rozdíl od anorganických scintilátorů v nich nedochází k excitaci jednoho atomu v krystalové mřížce, ale k excitaci celých molekul. Tuto vlastnost mají nejčastěji molekuly s aromatickým jádrem, jelikož právě uvnitř tohoto uhlíkového kruhu dochází k excitaci a deexcitaci elektronů tvořících meziatomové vazby v aromatické molekule (viz. Obr. 1.1-1 vpravo).

Z organických scintilátorů se nejčastěji používá naftalen, který vyzařuje scintilační záření o velmi krátké době záblesku (0,08ms) a vlnové délce kolem 345nm. Další důležitou organickou scintilační látkou je anthracen, který se používá jako normál

pro porovnávání vlastností všech ostatních organických scintilátorů. Anthracen emituje scintilace s dobou záblesku 0,03ms a vlnovou délkou 450nm a jeho konverzní účinnost je asi poloviční než u NaI(Tl). Z dalších organických látek můžeme uvést ještě stilben, který emituje scintilace s dobou trvání 0,08ms, vlnovou délkou 380-410nm a jeho výhodou je možnost vytvářet velké krystaly. Pro detekci záření gama mají organické scintilátory příliš malou hustotu, takže detekční účinnost bývá nízká. Jsou však velmi vhodné pro detekci elektronů β, částic α, protonů, deuteronů a rovněž i rychlých neutronů.

V anorganických scintilátorech je scintilační efekt vlastností vhodně uspořádané krystalové mřížky s luminiscenčními centry. Při rozpuštění anorganické látky (např. ve vodě) krystalová mřížka zaniká a scintilační efekt mizí.

Na druhou stanu, jak již bylo zmíněno, v organických scintilátorech vznikají scintilace deexcitací vlastních molekul vhodných organických sloučenin. Tudíž při rozpuštění takového scintilátoru ve vhodném organickém rozpouštědle (toluen, xylen, benzen, dioxan, fenyl-cyklo-hexan, fenyléter apod.), ve kterém nedojde k rozpadu těchto molekul, je scintilační efekt je většinou zachován. Tímto způsobem je možno vytvořit kapalný (tekutý) scintilátor.

Scintilátory mají několik základnách vlastností, které jsou důležité pro jejich použití. Základním parametrem je potom **Konverzní účinnost**, což je vlastně podíl energie, která ze scintilátoru vystupuje ve formě emitovaného světla a energie vstupního ionizujícího záření. Nutno dodat, že i u nejlepších scintilačních detektorů je tato účinnost poměrně malá. Další důležitou vlastností scintilátoru je jeho **Scintilační spektrum**, které udává vlnové délky emitovaného scintilačního záření. Toto spektrum je především důležité pro výběr fotokatody, která musí mít v této spektrální oblasti své maximum spektrální citlivosti. Viz kapitola 1.2.2. *(Ullman, 2005)*.

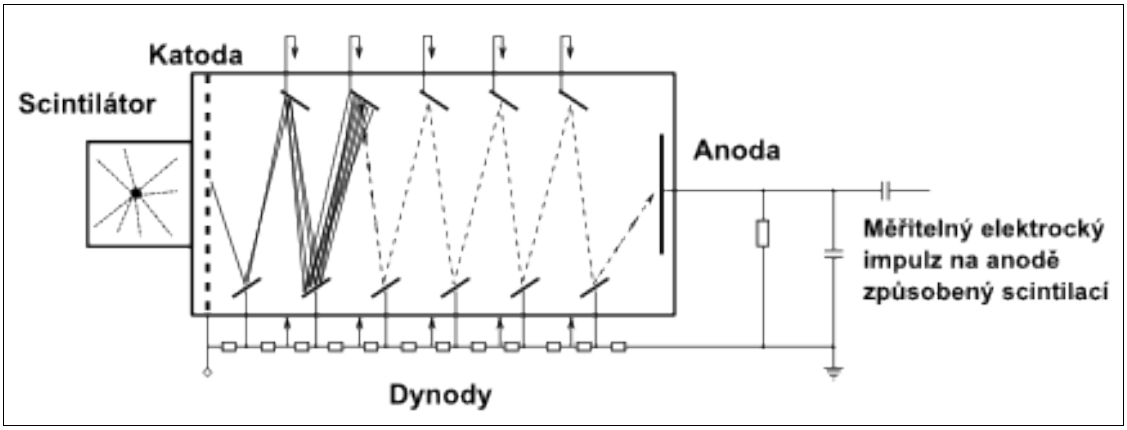
1.2 Fotonásobič

Fotonásobič je speciální elektronka, která s vysokou citlivostí převádí světlo na elektrický signál a která je opticky spojena se scintilátorem. Aby bylo možno detekovat i jeden jediný foton, který na fotonásobič dopadne, je malý počet elektronů, které

vzniknou na vstupu fotonásobiče z fotonů ze scintilátoru díky fotoelektrickému jevu, několikrát znásoben při průchodu elektronu skrz elektronku vyrážením sekundárních elektronů. Výsledkem by na výstupu fotonásobiče měl být dobře detekovaný elektrický pulz. Prvotní malé množství fotonů ze scintilátoru je díky fotonásobiči zvětšeno 10^6 - 10^7 krát.

Fotonásobiče se používají nejen ve scintilačních detektorech, ale i ve spektrofotometrii, detekci luminiscence (fyzikálního, chemického či biologického původu), detekci Čerenkovova záření a v dalších technických aplikacích. Klasické fotonásobiče jsou speciální vakuové elektronky, v nichž jsou elektrony generovány fotoemisí z fotokatody. Mezi anodu a katodu je přivedeno napětí okolo 1000 V.

Fotokatoda musí být dostatečně tenká, aby elektrony uvolněné foto-efektem mohly snadno vylétnout ven. Materiálem fotokatody jsou látky s nízkou výstupní prací elektronů pro fotoefekt. Nejčastěji jsou to antimonidy alkalických prvků, např. cesia a antimonu, bialkalické materiály, dále Ag-O-Cs, nebo multialkalické Na-K-Sb-Cs. Jak jsme si již řekli, nutné aby maximum spektrální citlivosti této fotokatody leželo někde ve středu spektra vycházejícího ze scintilátoru, abychom dostali maximální zesílení.



Obr. 1.2.1 Schéma fotokatody

Aby docházelo k zesílení, je mezi katodu a anodu vložena soustava elektrod, které mají oproti katodě kladný náboj a říká jim dynody. Napětí na dynodách se postupně zvyšuje, díky odporové matici.

Povrch dynod je tvořen tenkou vrstvičkou kovu s nízkou výstupní prací elektronu

(nejčastěji SbCs nebo BeO) a tím vysokým součinitelem sekundární emise, podobně jako je materiál fotokatod. Po dopadu elektronu na dynodu dojde k uvolnění dalších elektronů z jeho povrchu ve větším množství. Násobící systém, sestávající z cca 8-12 dynod, je zakončen sběrnou elektrodou (anodou) s nejvyšším kladným potenciálem, odkud se přes pracovní odpor odebírá výstupní signál.

Velikosti, provedení a uspořádání dynod se mohou velmi lišit a je známo několik dalších možností konstrukce fotonásobiče s kruhově uspořádanými dynodami či s dynodami ve lamelového provedení *(Ullman, 2005)*.

1.3 Zpracování výstupních impulsů ze scintilačního detektoru

Při detekci záření se na výstupu fotonásobiče objevuje velké množství elektrických impulsů různých velikostí, které je nutno dále zpracovávat a vyhodnocovat. K tomu slouží příslušné elektronické obvody, které impulsy zesilují, třídí je podle velikosti, počítají, registrují a zobrazují.

Výstupní signál z fotonásobiče je třeba nejprve zesílit v předzesilovači, který také upraví amplitudu eklektického proudu tak, aby byla úměrná počtu světelných fotonů dopadajících na fotokatodu. Tím by měl být i počet fotonů úměrný energii fotonů ionizačního záření interagujících se scintilačním krystalem.

Impulzy jsou dále zesilovány pomocí zesilovače a zpracování těchto impulsů lze provádět dvěma základními způsoby, podle účelu scintilačního měření (horní a dolní větev ve schématu na Obr. 1-1).

Pokud chceme měřit pouze počet impulzů, které mohou odpovídat třeba počtu rozpadů radioaktivních částic ve vzorku, tak stačí impulsy ze zesilovače dovést na amplitudový analyzátor. Amplitudový analyzátor je elektronický obvod propouštějící impulsy, jejichž amplituda leží v nastavitelném rozmezí mezi dolní a horní diskriminační hladinou. Impulsy ležící mimo toto rozmezí nejsou propuštěny. Nastavení dolní a horní hladiny se děje nezávisle, nebo častěji se nastavuje základní úroveň a kolem ní "okénko".

Okénko analyzátoru se nastavuje většinou tak, aby zahrnovalo fotopík ionizačního

záření měřeného radionuklidu. Normované impulsy z analyzátoru se již vedou do čitače. Naměřený počet impulsů za zvolený časový interval, je pak úměrný počtu impulsů záření gama *(Ullman, 2005).*

Tento systém používá i scintilační počítač Beckman LS 1801, kdy se na jeho výstupu objeví hodnota CPM, což je zkratka pro „count per minute“, čili impulzů za minutu. Tato hodnota odpovídá záblesků, které detektor zaznamenal ve vzorku za jednu minutu. Vzhledem k tomu, že jak scintilátor, tak fotonásobič, nedokáží znamenat a patřičně zesílit každý dopad ionizujícího záření, (ať již kvůli konverzní účinnosti scintilátoru nebo konstrukční nedokonalostí fotonásobiče), je tato hodnota obvykle vydělena ještě konstantou účinnosti, čím dostaneme hodnotu DPM (disintegration per minute), čili rozpadů za minutu, která by měla odpovídat skutečné aktivitě vzorku.

Další možností, jak vyhodnocovat impulzy ze scintilačního detektoru, je spektrometrická analýza. Impulsy ze zesilovače vedeme na analogově-digitální převodník, kde se analogová velikost amplitudy impulsu převádí na digitální bitovou kombinaci. V paměti počítače, či analyzátoru pak postupně vzniká digitální energetické spektrum, kde adresa každé paměťové buňky je úměrná energii záření gama a její nastřádaný obsah představuje detekovaný počet fotonů této energie. Při zobrazení na obrazovce počítače pak dostaneme typickou křivku scintilačního spektra (Obr. 1-1).

1.4 Detekce beta záření

V minulých částech jsem se zabýval především způsoby detekce γ záření. V laboratorní technice se ale velmi často používají také β^- zářiče. Scintilační počítač Beckman LS 1801 je také především určen pro detekci β^- záření.

Na rozdíl od γ záření, které je podobně jako rentgenové záření elektromagnetické vlnění je záření β^- tvořeno proudem nabitých částic, konkrétně elektronů. Toto záření má sice mnohem vyšší ionizační účinky, než záření γ, ale také mnohem nižší pronikavost. Beta záření například nepronikne ani normálním oblečením a dělá mu problém i průnik lidskou kůží. Dobře totiž interaguje s hmotou, kdy dochází k záchytu elektronu, který tvoří záření β^-, nejbližším atomem látky, která zářič obklopuje.

Pro použití β^- zářičů v laboratorní i jiné měřící technice hovoří několik výhodných

vlastností. Jak jsme si již řekli, tyto zářiče nejsou pro člověka tak nebezpečné z hlediska vnějšího ozáření a mezi β^- zářiče patří několik izotopů, které mají unikátní vlastnosti.

Nejčastěji používané β^- zářiče jsou izotopy ^{3}H (tricium), ^{14}C, ^{32}P a ^{125}I. Jsou to všechno prvky hojně se vyskytující v organických strukturách a tudíž jsou vhodné pro tvorbu indikátorů (markerů) při nejrůznějších měřeních. Tricium je jako izotop vodíku jedním z nejpoužívanějších, neboť skoro každá organická molekula obsahuje vodík.

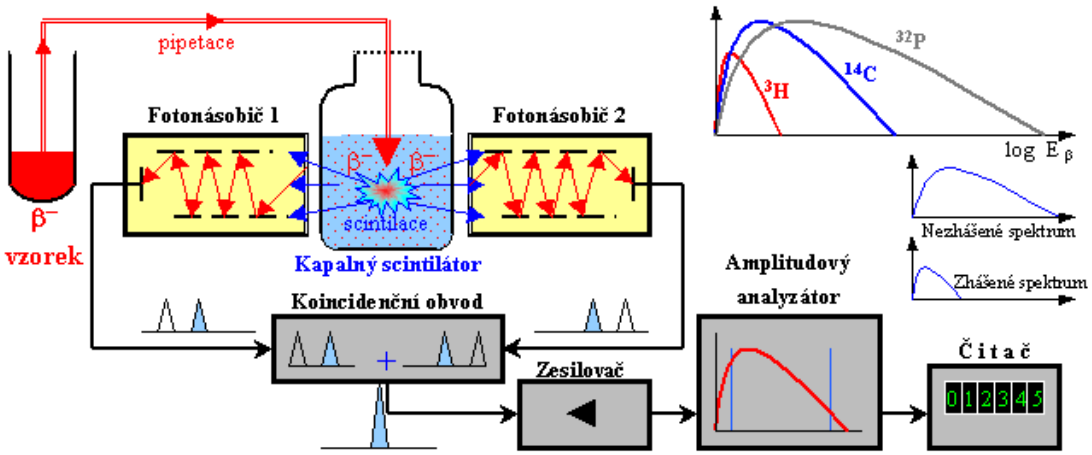
V laboratoři neurofyziologie je používáno právě tricium a jako scintilátor naftalen v roztoku dioxanu.

Tyto izotopy však mají i své mínusy, především pak velmi nízkou energii. Z hlediska detekce záření β^- je tu problém s vysokou absorpcí částic přímo v měřením vzorku, ještě než toto záření dopadne na detektor. Malé množství elektronů, které by sice dokázaly opustit měřený vzorek, by se jistě absorbovalo na obalu vzorku (např. skleněná zkumavka) a ve výsledku by na detektor nedopadalo vůbec žádné β^- záření.

Jistou možností je měřit záření na velmi tenké vrstvičce vzorku, avšak i tady je při použití klasických scintilátorů na organické bázi problém s konverzní účinností detekce, zvláště pak u nízkoenergetického záření β^-. Tady ani sebedokonalejší detektory záření s krystalickým scintilátorem nejsou ve výsledku dostatečně přesné, aby se dali spolehlivě použít pro měření.

Existuje však řešení, které velmi zvyšuje konverzní účinnost detekce, která je ve výsledku mnohem vyšší než u klasických krystalických scintilátorů. Tímto řešením je použití kapalných scintilátorů. Jak jsme si již řekli, tak některé organické scintilátory neztratí svoji schopnost scintilace, pokud je rozpustíme ve vhodném rozpouštědle.

Měřený vzorek přimícháme přímo do roztoku kapalného scintilátoru, nebo scintilační roztok přidáme ke vzorku v dostatečném množství. Vzniklou směs potom musíme umístit do průhledné zkumavky. Tak zcela odstraníme problém absorpce záření β^- přímo ve vzorku, neboť každý zářič β^- je ze všech stran obklopen molekulami scintilátoru. Při radioaktivní přeměně tedy bude vylétající elektron bezprostředně interagovat se scintilátorem a tím vznikne scintilační záblesk.



Obr. 1.4-1 Schéma detekce záření β a principu fungování scintilačního detektoru, který používá kapalný scintilátor. [2]

Takto vzniklé scintilace v kapalném scintilátoru jsou snímány fotonásobiči, kde se světelné záblesky převádějí na elektrické impulsy stejně jako je tomu u běžných scintilačních detektorů. Amplituda výstupních impulsů je úměrná intenzitě scintilace, a tedy energii detekovaných elektronů. Dále pak počet zaregistrovaných impulsů za jednotku času je úměrný aktivitě preparátu.

Při detekci β^- záření se často používá zapojení dvou fotonásobičů současně v takzvaném koincidenčním zapojení. Toto zapojení umožňuje zvýšení účinnosti detekce a také potlačení nežádoucích impulsů pocházejících z šumů fotonásobiče a z chemiluminiscence.

Některé chemické reakce mezi materiálem vzorku a scintilátorem mohou také vést k emisi světla, podobně jako scintilace. Těmto jevům říkáme chemiluminisce a na rozdíl od scintilace nemá původ v detekovaném záření β^- a trvá déle. Chemiluminiscence je oproti scintilaci také rozdílná v tom, že při jednom aktu se vyzařuje jen málo fotonů, zatímco scintilace je zábleskem několika stovek nebo tisíc fotonů. Použijeme-li tedy pro detekci světla z kapalného scintilátoru dvou fotonásobičů (viz Obr. 1.4-1), pak šum i fotony z chemiluminiscence vyvolají impuls nezávisle vždy jen v jednom z fotonásobičů. Na druhou stranu velké množství současně emitovaných fotonů při scintilaci dopadne současně do obou fotonásobičů.

Na výstupu z koincidenčního obvodu tedy dostáváme impulsy jen tehdy, když je detekována scintilace vyvolaná zářením β^-, zatímco šum a chemiluminiscenční impulsy

propuštěny nejsou. Toto potlačení nežádoucích impulsů má význam zvláště při měření vzorků tricia, kdy šum z fotonásobiče má amplitudu srovnatelnou se signálními impulsy, pocházejícími z detekce nízkoenergetického záření β^- *(Ullman, 2005).*

Celý detektor s fotonásobiči je samozřejmě uzavřen ve světlotěsném stíněném boxu, kam se zkumavka, zvaná scintilačka (viz Obr. 1.4-3), s měřeným vzorkem, smíchaným s kapalným scintilátorem, spouští přes světelnou clonku pomocí elevátoru a po proběhnutí měření se vysouvá zpět.

Beckman LS 1801 (viz Obr. 1.4-2) je vybaven mechanickým systémem vzorkovnic o kapacitě několika desítek až stovek scintilaček,. Ty jsou postupně zasouvány do detektoru, a po uplynutí předvolené měřící doby jsou zase vysouvány.

Obr. 1.4-2 Scintilační počítač Beckman LS 1801 ve Fyziologickém ústavu 1. Lékařské fakulty Univerzity Karlovy v Praze.

Obr. 1.4-3 Scintilační zkumavky tzv. „scintilačky“ ve vzorkovnici.

2 Sériový port

Sériový port (často označovaný jako RS-232), byl původně koncipován pro připojení textových terminálů k jednoduchému modemu. To se samozřejmě posléze změnilo a pomocí tohoto rozhraní bylo k počítači připojováno mnoho jiných zařízení. Namátkou je třeba zmínit světelná pera, počítačové myši, plottery a některé tiskárny, původní analogové i moderní vysokorychlostní modemy a propojovaly se s ním počítače, budovaly jednoduché počítačové sítě. Jak takový konektor sériového portu vypadá se ukazuje Obr. 2-1.

Obr. 2-1 Sériový port - zde konkrétně konektor Canon 9pin Male na zadní straně přístroje pro testování pulzních oxymetrů

Dnes už je tento typ komunikačního rozhraní technicky dávno překonán, ale i tak ho neustále můžeme najít na nejrůznějších zařízeních (moderních strojích jako je chirurgický robot, některé servery, průmyslové stroje atd). Takže ať se zdá, že tento koncept je jich hodně zastaralý, můžeme se s ním setkat i na místech, které bychom nečekali.

Pro počítače, které by potřebovaly tento port, ale nemají ho (poměrně výrazně to zvyšuje cenu základní desky), je možné dokoupit konvertor mezi universální sériovou sběrnicí (USB) a sériovým portem, či mezi kartou typu PCMCIA a RS-232. Právě převodník mezi USB a RS-223C od firmy PremiumCord je využíván pro připojení scintilačního počítače Beckman Counter 1801 k PC, kde běží program Boreas.

2.1 Sériová komunikace

Pokud se bavíme o sériové komunikaci, znamená to, že se data, které potřebujeme dostat na druhou stranu kabelu, neposouvají celé najednou, nýbrž se rozdělí na menší části a ty se pošlou postupně.

Sériový přenos dat je velmi stará záležitost, která tu byla již před příchodem počítačů. Typickým příkladem je telegraf. Telegrafista na jedné straně může pouze vysílat data a druhý telegrafista je může pouze přijímat. V tomto případě si vystačíme s jediným kabelem datovým a druhým pro společný zemnící vodič. Takový způsob komunikace se nazývá simplexní. Pokud bychom nechtěli přidávat další kabel, ale současně potřebujeme komunikovat oběma směry, musíme si vytvořit nějaké pravidla. Takže pokud bude jedna strana vysílat data, druhá bude poslouchat. První strana ukončí komunikaci vysláním speciálního znaku a tím řekne: „*Už jsem se vypovídal, řada je na tobě*“. Strany se prohodí a druhá strana začne vysílat data, zatímco první bude zase pouze poslouchat. Tento způsob je v technice poměrně častý, používá se u desetimegabitového i stomegabitového Ethernetu, a nazývá se poloduplexní přenos dat. Pokud bychom chtěli současně data přijímat i vysílat, potřebujeme další vodič. Takový typ komunikace se nazývá plně duplexní (fullduplex) přenos. I tento typ komunikace se u telegrafu používal a zasadil se o něj třeba i Thomas A. Edison. Podobný způsob zapojení - dvojice vodičů TxD (Transmit Data) → RxD (Receive Data) a RxD → TxD doplněná o společný zemnící vodič (Ground) – se používá i u sériového portu *(Tišnovský, 2008).*

2.2 Komunikace po sériovém portu

Sériový port typu RS-232 patří do rodiny zařízení nazývaných UART, což je zkratka znamenající Universal Asynchronous Receiver and Transmitter, neboli universální asynchronní přijímač a vysílač. V tomto názvu je důležité slovo asynchronní, protože naznačuje, jakým způsobem jsou data rozložená na proud bitů přenášena.

Jelikož při sériové komunikaci data následují za sebou, musí přijímací zařízení a vysílací zařízení být synchronizováno, jinak by se od sebe jednotlivé byty nedaly

rozlišit. Pokud by přijímač nevěděl, kdy vysílač vysílá, nevěděl by ani, kdy má číst z proudu dat. Pro tento účel se buď kromě vlastního datového vodiče používá ještě jeden vodič s hodinovým signálem (synchronní přenos), nebo se informace o synchronizaci vhodným způsobem zkombinuje s proudem přenášených bitů.

Asynchronní přenos je typický tím, že se v samotných přenášených datech informace o synchronizaci neobjevují, a nepoužívá se ani samostatný synchronizační signál. To by však znamenalo, že přijímací i vysílací strana musí používat naprosto přesný zdroj hodinového signálu použitý pro detekci hranic jednotlivých bitů, což v praxi není možné, protože i komerčně dostupné krystaly, běžně považované za velmi přesné, by již po několika sekundách přenosu měly takovou odchylku, že by přesáhla dobu trvání přenosu jednoho bitu. Proto se i u přenosových zařízení UART synchronizace používá, ale ne na úrovni jednotlivých bitů.

Sériový port, stejně jako každé zařízení typu UART, vkládá do proudu dat synchronizační značky. Vetšinou to je na začátku a na konci nějakého většího počtu bitů, typicky potom 8, takže okolo jednoho bajtu.

Aby takovýto přenos byl vůbec proveditelný, musí si obě stany nastavit několik parametrů komunikace. Jedná se především o formát data a rychlost vzorkování datové linky. K tomto slouží následující hodnoty:

Baud je modulační rychlost, což je v podstatě počet změn stavu přenosového média za jednu sekundu a přestavuje rychlost komunikace. Je nutné aby oba přístroje, které jsou sériovou linkou propojeny, komunikovaly stejnou rychlostí. Hodnoty jsou typicky násobky 300 a hodnoty sahají od 300 do 921600 viz Tabulka 2.2-1.

Tab. 2.2.1 Tabulka přenosových rychlostí sériového portu [4]

Baud	Doba přenosu jednoho bitu	Doba přenosu jednoho bajtu
1200	833 µs	8,33 ms
2400	416 µs	4,16 ms
4800	208 µs	2,08 ms
9600	104 µs	1,04 ms
19200	52 µs	520 µs
38400	26 µs	260 µs
115200	8,6 µs	86 µs

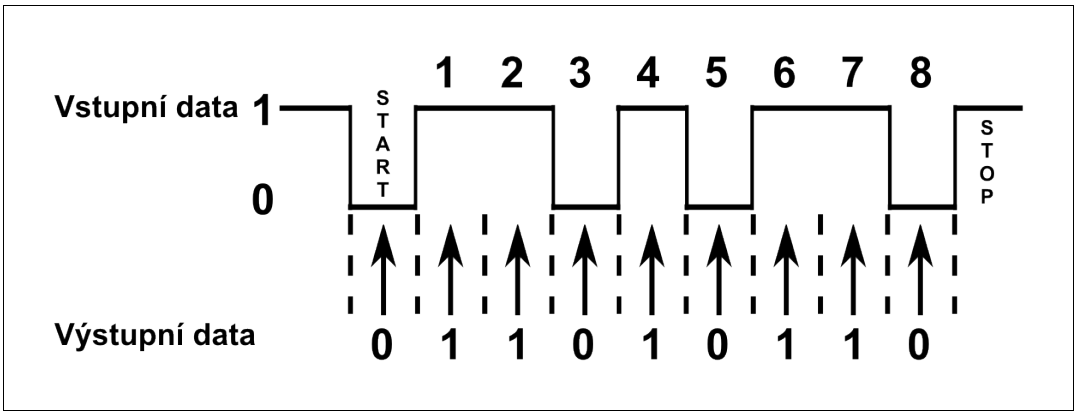
Počet datových bitů. Jedná se o počet bitů, které se odešlou v jednom bloku sériového přenosu. Nejčastější hodnoty jsou 5, 6, 7 a 8. Úplně nejčastější hodnota je právě 8 bitů.

Paritní bit je v podstatě druh kontrolního součtu. Paritní bit je určen k jednoduché detekci chyby v přenosu. Pomocí paritního bitu lze detekovat lichý počet chyb ve slově. Lichá parita znamená lichý počet jedničkových bitů ve slově (i s paritním bitem), sudá parita sudý počet jedničkových bitů ve slově. Možné hodnoty jsou none, odd, even, mark a space. None znamená, že tento mechanizmus bude vypnut a žádný paritní byt se nepřenáší. Odd je lichá a Even sudá parita. Mark potom znamená, že paritní byt je vždy roven 1 a Space potom 0. Nutno dodat, že poslední dvě hodnoty se příliš nepoužívaly ani v minulosti.

Délka stop bitu. Stop bit je bit určující konec jednoho vysílaného slova. Tento specifický bit má vždy hodnotu 1 a může být různě dlouhý, od 1, 1,5 až po dvojnásobek délky datového bitu.

Logický stav 0, anebo 1 přenášených dat je reprezentován pomocí dvou možných úrovní napětí, které jsou bipolární a dle typu zařízení mohou nabývat hodnot ±5 V, ±10 V, ±12 V nebo ±15 V. Nejčastěji se používá varianta, při které logické hodnotě 0 odpovídá napětí −12 V a logické hodnotě 1 pak +12 V. Právě tyto vysoké hodnoty napětí, jsou jedním z důvodů, proč se dnes již z cenových důvodů na základní desky počítačů tento port neinstaluje.

Na začátku komunikace se datový vodič nachází ve stavu logická 1, což je stav vysokého napětí. Nejprve je poslán takzvaný start bit, který má vždy nulovou hodnotu, takže klidový stav linky se vždy změní. Na straně přijímacího zařízení se musí co nejpřesněji rozeznat změna stavu linky, která reprezentuje začátek start bitu. Od této chvíle přijímací strana ví, že se přijme předem daný počet bitů s předem nastavenou bitovou rychlostí. Přijímací strana tedy vzorkuje stav linky a podle svých vlastních hodin rozeznává hodnoty jednotlivých bitů. Na konci přenášené sekvence může být přenesen i paritní bit, za nímž ihned následuje stop bit. Stop bit má zaručit, že přijímač má dostatek času na zpracování došlých dat. Viz Obr 2.2-1

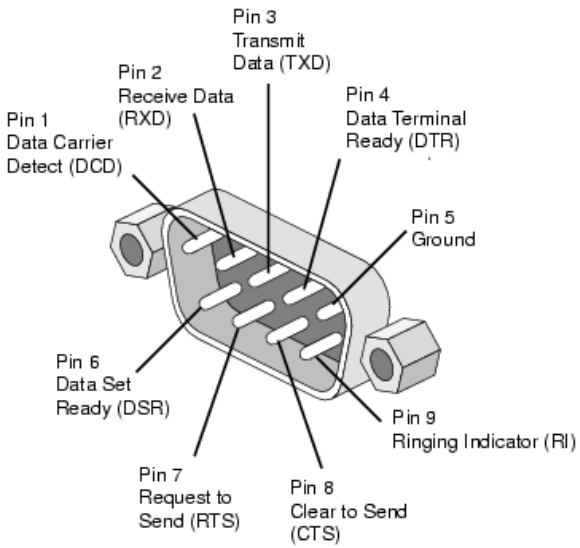


Obr. 2.2-1 Asynchronní přenos jednoho bajtu (osmi bitů) po sériové lince. Šipky označují místa, kde dochází ke čtení dat přijímačem.

Díky základnímu stavu na začátku komunikace, hodnotě start a a stop bitu, je zajištěno , že vždy dojde ke změně stavu na lince, i když se budou přenášet data pouze hodnoty 1 nebo 0. Samotné čtení dat na straně přijímače se provádí vzorkováním signálu, který přichází po lince. Například u některých sériových portů použitých v osobních počítačích je vzorkovací frekvence 16× vyšší než baud, což znamená, že v průběhu trvání jednoho bitu se ve skutečnosti přečte šestnáct vzorků, které se dále zpracovávají takovým způsobem, aby se vyloučilo rušení, nevyvážená přenosová rychlost, případné chyby atd.. Typicky se v úvahu berou hodnoty vzorku 8, 9 a 10, ze který se vypočítá hodnota, která se bere jako definitivní. Na ideální lince, by samozřejmě nedocházelo k rušení a všech šestnáct hodnot by bylo stejných. Vysoká vzorkovací frekvence je nutná především pro přesné určení start bitu, protože od toho se v přijímači odvozují veškeré časy začátku dalších bitů *(Tišnovský, 2008)*.

2.3 Zapojení konektorů a kontrola toku dat

Klasický konektor DB-9/DE-9 (viz Obrázek 2-1), má k dispozici 9 pinů, což ale nemusí odpovídat devíti připojeným „drátům“. Různé piny jsou specificky přiřazené k určité funkci (viz Obrázek 2.3-1). Pro mnoho zařízení, připojených k osobnímu počítači postačují pouze piny TxD, RxD a Ground. Další piny slouží především k hardwarovému zajištění toku dat (hardwarový handshaking). Jedná se o dvojce pinů DTR/DSR a RTS/CTS. Poslední dva piny jsou na rozdíl od handshakingu dnes již zcela nepoužívané. DCD je označení pro pin určený pro detekci nosné vlny. Modem tímto signálem oznamuje terminálu, že na telefonní lince detekoval nosný kmitočet. RI je potom Ring Indicator. To lze přeložit jako indikátor zvonění. Modem tímto signálem oznamuje terminálu, že na telefonní lince detekoval signál zvonění *(Olmr, 2005)*.



Obr. 2.3-1 Popis jednotlivých pinů konektoru DB-9/DE-9. Popis odpovídá zapojení použitém při komunikaci po sériovém portu. [6]

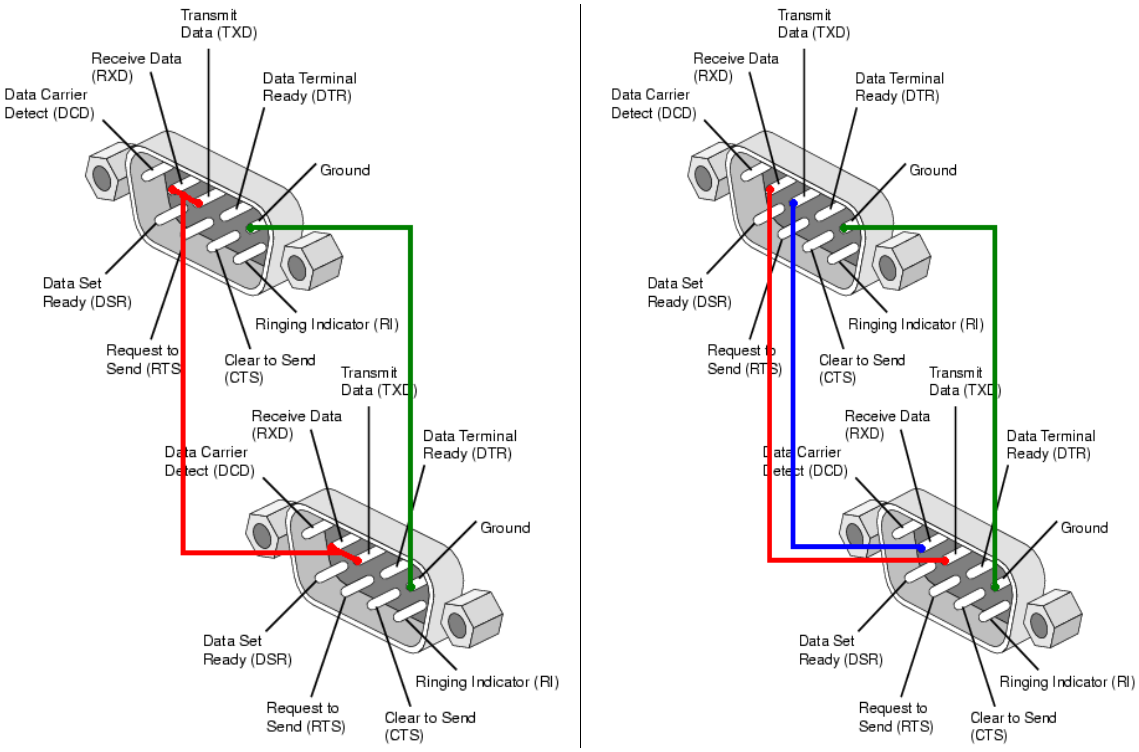
Kromě klasického DB-9/DE-9 konektoru, je možno narazit i na DB-25/DE-25 konektor s 25 piny. Šestnáct dalších pinů mohlo být použito ke zvýšení přenosové kapacity sériové komunikace, nebo mohlo mít jiný účel podle specifikace výrobce.

Dále se budu zabývat jednotlivými možnostmi zapojení linky, podle použitého typu řízení přenosu dat (handshakingu).

2.3.1 Softwarové řízení přenosu dat

V případě, že využíváme poloduplexní přenos dat, stačí nám zapojit pouze dva vodiče, kdy prvním vodičem spojíme vzájemné piny Ground a druhý vodič slouží k přenosu dat, a to směrem tam i zpět.

Na obr. 2.3.1-1 vlevo je ukázáno, jakým způsobem jsou obě komunikující strany navzájem propojeny. Všimněte si, že na každém konci přenosové linky, jsou spojeny piny *RxD* a *TxD*. Použití pouhých dvou vodičů při přenosu je umožněno tím, že vždy jedna strana vystupuje jako vysílač a druhá strana jako přijímač, přičemž na prohození obou rolí se musí obě komunikující strany dohodnout pomocí protokolu.



Obr. 2.3.1-1 Způsob zapojení sériového portu při poloduplexním (vlevo) a plně duplexním přenosu dat (vpravo) při použití softwarového handshakingu. [7]

V současnosti se ve většině případů používá plně duplexní přenos, při kterém jsou obě zařízení propojena dvěma datovými vodiči zapojenými podle schématu na Obr 2.1-1 vpravo. V tomto případě se zařízení nemusí ohlížet na druhé zařízení a může vysílat a přijímat data nezávisle.

Na první pohled by se mohlo zdát, že v tomto případě není vůbec zapotřebí přenos

dat nějakým způsobem řídit, ale ve skutečnosti je nutné zajistit, aby spolu bez ztráty informací mohly komunikovat i ta zařízení, která data zpracovávají rozdílnou rychlostí.

V případě, že modem již nestíhá data dodávaná počítačem zpracovávat, protože má naplněnou vyrovnávací paměť, může v případě softwarového hangshakingu vyslat zpět do počítače ASCII znak s kódem 19, který je také známý svým označením X-OFF. Počítač tento znak přijme a ihned poté přestane odesílat další data. Ve chvíli, kdy je modem opět připraven data přijímat, vyšle po sériové lince ASCII znak s kódem 17, který se též označuje jako X-ON. Právě tato jednoduchá metoda řízení toku dat se označuje jako X-ON/X-OFF nebo také softwarový handshaking, nebo také **software flow control** *(Tišnovský, 2008).*

2.3.2 Hardwarové řízení přenosu dat

V praxi se ustálily dva způsoby hardwarového řízení. První způsob využívá pinů **RTS** a **CTS**, druhý způsob pak pinů **DTR** a **DSR**. Popis těchto pinů je znázorněn v Tabulce 2.3.2-1

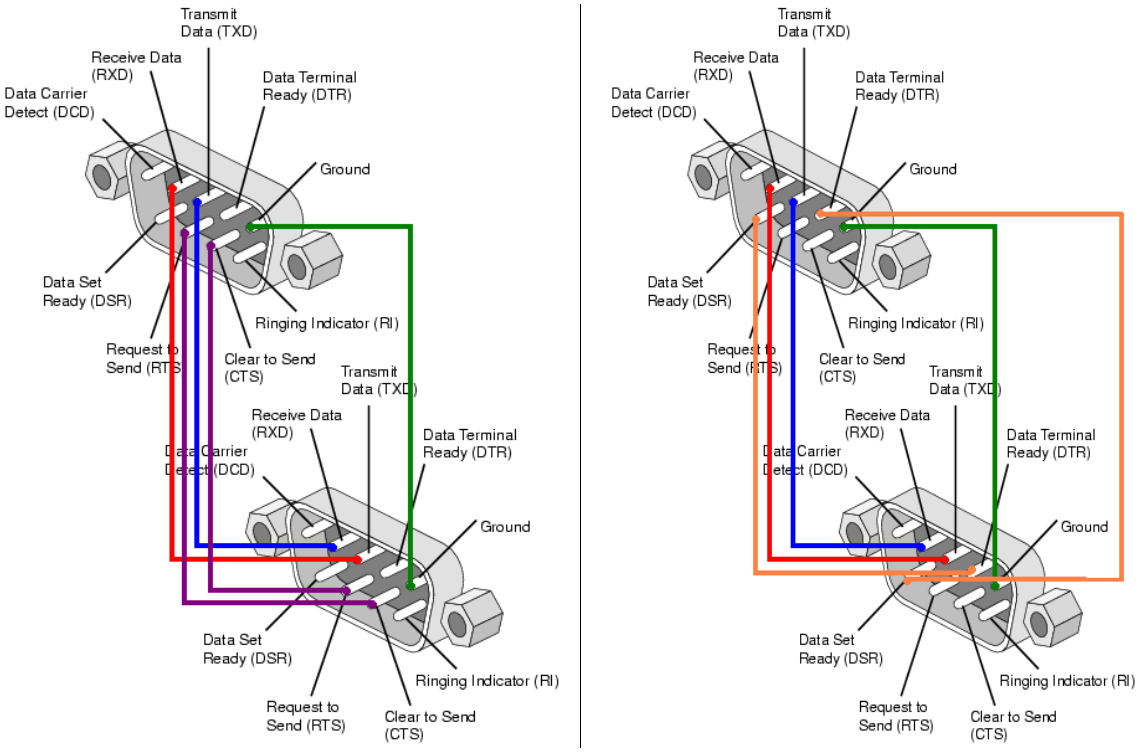
Tab. 2.3.2.1 – Název a popis pinů užívaných v hardwarovém handshakingu [3]

Signál	Popis
DTR - Data Terminal Ready	Terminál tímto signálem oznamuje modemu, že je připraven komunikovat
DSR - Data Set Ready	Modem tímto signálem oznamuje terminálu, že je připraven komunikovat
RTS - Request to Send	Terminál tímto signálem oznamuje modemu, že komunikační cesta je volná
CTS - Clear to Send	Modem tímto signálem oznamuje terminálu, že komunikační cesta je volná

Signály plní funkci “přepínačů” pro řízení poloduplexní komunikace. V případě komunikace po sériovém portu není zapotřebí potvrzovat příjem každého bitu, jelikož tato činnost je zajištěna samotným čipem UARTu. Zařízení tedy pomocí signálů dává najevo, že již může nebo nemůže přijmout další bajt (znak). Signálem se myslí zvýšení napětí na příslušném pinu na úroveň logická 1.

V plně duplexních komunikačních zařízení ztrácejí řídící signály částečně svůj

původní význam a programy je využívají spíše způsobem *"Terminál hlásí, že je momentálně připraven (nepřipraven) přijmout data"*. K tomu může programátor využít jak signál DTR, tak signál RTS.



Obr. 2.3.2-1 *Způsob propojení dvou zařízení při komunikaci přes sériový port při hardwarovém řízení RTS/CTS (vlevo) a při použití DTR/DSR (vpravo). [7]*

Při hardwarovém řízení známém také pod označením RTS/CTS, jsou potvrzovací signály posílané mezi dvojicí pinů RTS a CTS. Řízení však musí být v případě plně duplexního přenosu dat prováděno obousměrně, proto se používá způsob zapojení zobrazený na Obr. 2.3.3-1. Všimněte si, že piny RTS a CTS jsou zapojeny tak, že vedou vždy na druhý pin, než jsou oni samy.

Komunikaci lze řídit také i přes dvojce pinu DRT/DSR a tento způsob řízení je v podstatě stejný jako RTS/CTS, jen se pro něj využívá jiných pinů. Dnešní kabely sériového portu mají obvykle zapojeny oba kontrolní mechanizmy, ale častější se používá RTS/CTS handshake *(Tišnovský, 2008)*.

2.4 Délka kabelu sériového portu

Standard RS 232 uvádí jako maximální možnou délku vodičů 15 metrů, nebo délku vodiče o kapacitě 2500 pF. To znamená, že při použití kvalitních vodičů lze dodržet standard a při zachování jmenovité kapacity prodloužit vzdálenost až na cca 50 metrů. *(Olmr, 2005).*

Tab. 2.4-1 Maximální délka kabelu sériového portu v závislosti na přenosové rychlost [4]

Baud rate	**Maximální délka v metrech**
2400	900
4800	300
9600	150
19200	15

Texas Instruments uvádí tyto délky vodičů jako výsledek pokusných měření. Vzhledem k „laboratorním“ podmínkách tohoto měření je třeba brát tyto údaje pouze jako orientační. V praxi je třeba počítat s rušením atd.

Kabel lze také prodlužovat při snížení přenosové rychlosti, protože potom bude přenos odolnější vůči velké kapacitě vedení. Výše uvedené parametry počítají s přenosovou rychlostí 19200 Bd a pro některé další hodnoty se podívejte do Tabulky 2.4-1.

2.5 Sériová komunikace v operačních systémech

V případě, že zařízení připojené k sériovému portu komunikuje standardním způsobem, tedy pro posílání a příjem dat se používá asynchronní způsob přenosu po pinech TxD a RxD a řízení toku dat je prováděno buď softwarově (X-ON, X-OFF) nebo hardwarově (RTS/CTS, DTR/DSR), není většinou nutné vytvářet vlastní ovladač, který by sériový port přímo řídil pomocí jeho konfiguračních registrů. Místo toho se lze na sériový port dívat jako na běžné znakové zařízení, na které je možné vysílat jednotlivé znaky (bajty) a naopak číst znaky (bajty), které byly sériovým portem přijaty.

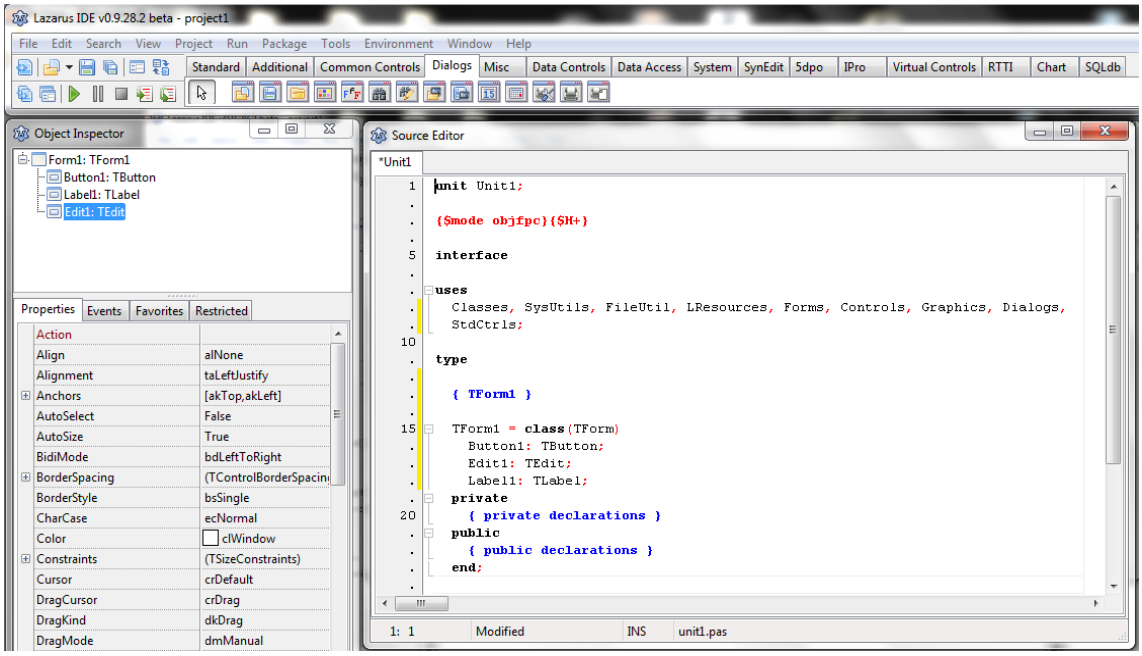
Většinou stačí správně nastavit parametry rozhraní (baud, start bit atd...), otevřít port a začít na port zapisovat data (znaky). V případě, že chceme data přijímat stačí v

pravidelných intervalech číst z portu znaky, které na něj zapsal vysílač.

Většina operačních systémů má včetně Linuxu, MS DOSu a starších verzí Windows přímo v sobě zakomponovanou podporu sériového portu a není potřeba pro jednodušší případy komunikace (pouze čtení nebo zápis jednoduchých dat), bez zvláštního protokolu, používat žádný dodatečný software *(Tišnovský, 2008)*.

3 Vývojové prostředí Lazarus

Program Boreas byl vytvořen ve vývojovém prostředí Lazarus (Obr. 3-1). Toto prostředí je multiplatformní, takže vývoj není omezen na použitý operační systém. Momentálně je podporován Microsoft Windows, různými distribucemi Linuxu, FreeBSD a Mac OS X. Další výhodou je filozofie programu, která zní "*Write once, compile anywhere!*" („*Napiš jednou, zkompiluj všude!*"). Takže pro program napsaný na Lazaru v Linuxu, se dá jednoduše vytvořit port pro Microsoft Windows tak, že se znovu zkompiluje v Lazaru nainstalovaném na počítači s Windows.



Obr. 3-1 Vývojové prostředí Lazarus běžící na operačním systému Windows 7

Dalším důvodem použití právě tohoto IDE, je fakt, že je uvolněn pod svobodnou licencí GPL a je k dipozici na všechny platformy zdarma. Navíc programy vytvořené tímto programem nejsou vázány GPL licencí a mohou být tedy šířeny libovolnou cestou, například i jako komerční produkt.

Programovacím jazykem v Lazaru je Pascal a Object Pascal a celé prostředí je inspirováno oblíbeným produktem Borland Delphi.

Lazarus zvládá i Delphi dialekt Pascalu a je možno do něj importovat některé projekty vytvořené v Delphi.

Celé IDE Lazarus je postaveno na FPC, což je svobodný překladač kódu pro Pascal a Object Pascal. Právě on umožňuje prování aplikací vytvořených v Lazaru na různé systémy a procesorové architektury, od i386 přes x86-64 až po ARM.

Programování v Pascalu a tvorbě aplikací v Lazaru se v našem se v našem studijním oboru věnuje předmět „*Základy algoritmizace a programování*“.

Jak již bylo řečeno, Lazarus je velmi podobný Delphi, zejména designu IDE a vnitřní filozofií. Také se jedná o RAD (Rapid Application Development – Rychlý vývoj aplikací) nástroj, který umožňuje vizuální návrh grafického uživatelského rozhraní, na jehož základě je automaticky vytvářena kostra zdrojového kódu, což výrazně urychluje vývoj celé aplikace. Lazarus přejal také koncept knihovny komponent, kdy dodávané komponenty významně usnadňují tvorbu aplikací a jsou sdružené v LCL (Lazarus Componet Libraly). Další komponenty lze stáhnout z internetu a někdy je možné použít i komponenty známé z Delphi. V Lazaru lze, stejně jako v Delphi, vytvářet vlastní komponenty.

Vývojové prostředí se podobně jako u Delphi skládá z několika oken. Důraz je zde kladen na vizuální programování, rychlost produkce programu a robustnost celého prostředí. Nutno říci, že obsáhlejší projekty naráží na přílišnou komplexnost a tak trochu „nepřehlednost“ celého IDE.

Nejdůležitější okna jsou Hlavní panel s Paletou komponent, Formulář, Objekt Inspektor a Code Editor.

3.1 Object Pascal

Programovací jazyk Object pascal vychází z programovacího jazyka Pascal. Ten byl vytvořem začátkem 70. let profesorem Niklausem Wirthem z Vysoké školy technické v Curychu. Název jazyka byl zvolen na počest francouzského filosofa, matematika a fyzika Blaise Pascala. Tento profesor chtěl pro své studenty vytvořit jazyk, jenž by byl vhodný pro výuku programování, a který by byl založen na omezeném počtu srozumitelných konstrukcí. Dále pak navrhl strukturu jazyka tak, aby bylo snadné implementovat Pascal na většině tehdejších počítačů.

První verze Pascalu byla publikována roce 1971. Využívání jazyka však odhalilo některé nedostatky, a proto byla roku 1974 uveřejněna opravená definice jazyka, která se stala téměř normou v počátcích jeho existence.

V dalším období narůstala nutnost vytvořit skutečnou normu jazyka, která by umožňovala jednoduchý přenos programů zapsaných v Pascalu mezi počítači. V roce 1981 byla vydána norma ISO. Vedle toho vznikla řada komerčních implementací Pascalu, přesněji řečeno dialektů Pascalu, které se od standardního Pascalu dle normy ISO více či méně odchylovaly, zejména zavedením dalších konstrukcí zjednodušujících praktické programování. V oblasti osobních počítačů dosáhla největšího úspěchu implementace firmy Borland pod názvem Turbo Pascal.

Popisovat zde všechny vlastnosti programovacího jazyka Pascal by bylo kontraproduktivní, takže si objasníme ty nejdůležitější.

Pascal má několik základních rysů: Je to **sekvenční** (to znamená, že provádí příkazy v pořadí, v jakém je napíšete do programu) **procedurální** (je postaven na procedurách a funkcích, což jsou jakési podprogramy, které jsou určeny pro konkrétní výpočet nebo činnost) **kompilovaný** (zdrojový kód programu není vykonáván přímo jako u skriptovacího jazyka, ale je přeložen do podoby spustitelného souboru, který může běžet nezávisle v operačním systému) programovací jazyk, který používá **deklarované** (proměnné nelze vytvářet v průběhu programu, ale musí být připravené již při jeho spuštění) silně **typové proměnné** (což znamená, že každé proměnná, které se v programu vyskytuje musí mít předem jasný typ, tedy jestli se jedná o textový řetězec, celé číslo, číslo s desetinou čárkou atd.).

Mezi další vlastnosti patří například fakt, že na rozdíl od jiných jazyků nerozlišuje v kódu velká a malá písmena. Např. proměnné *Auto, auto, AUTO* jsou podle překladače jedny a ty samé proměnné. Je však zaběhnutou praxí používat na začátku jména proměnné velké písmeno.

Důležitou vlastností Pascalu je možnost tvorby tzv. *unitů*, čili jednotek. To jsou soubory, ve kterých může být uložena část kódu a my ji můžeme přidat do našeho programu. Například můžeme mít někde funkci, která vypočítává faktoriál ze zadaného čísla. Pokud bychom vytvořili *unitu*, ve které by byla tato funkce nadefinovaná,

můžeme jí potom přidat k různým programům, které budeme vytvářet, bez toho, abychom ji museli vytvářet stále znovu. Typická *unita* má příponu .pas a může v ní být uložena funkce, konstanty, nebo v případě Object pascalu i deklarace tříd (viz. dále)

3.1.1 Objektově orientované programování

Object Pascal je rozšíření programovacího jazyka Pascal o některé vlastnosti objektově orientovaného programování. Co to je, se pokusíme v rychlosti vysvětlit.

Objektem se v Object Pascalu míní určitý soubor vlastností v jednom složeném datovém typu. Vlastnostmi daného objektu mohou být datové složky (proměnné), kterým se říká **atributy**, nebo procedury a funkce, zvané **metody**.

Chápání pojmu objekt v programování je zcela identické jako u objektů v reálném životě. Existuje-li nějaká konkrétní osoba, má své specifické datové složky, například *věk*, *pohlaví* a podobně. Umí-li dotyčná osoba sčítat, její metodou je *secti(a,b)*.

Jak bylo zmíněno, objekty se snaží simulovat realitu. Každý objekt je samostatný kus reality, který má jisté vlastnosti (atributy), můžeme s nimi dělat různé věci (metody) a objekty si samy pamatují svůj stav.

```
TPes = class // definice tridy
    Vyska: real;
    Vaha: real;
    Barva: TColor;
    procedure Sedni();
    procedure Lehni();
end;
```

Ukázka 3.1.1-1 Ukázka zdrojového kódu návrhu třídy TPes v programovacím jazyku Object Pascal

Důležité je, že každý objekt je potomkem (správně řečeno **instancí**) nějaké třídy. Například v programu by objekt *Pes* byl instancí třídy *TPes* (v Objeckt Pascalu je zvykem dávat před jméno třídy velké T). Můžeme mít dvě instance této třídy, třeba objekty *Alík* a *Rex*. Jsou to různí psi, takže budou mít různé hodnoty **atributů** jako je třeba *barva*, *váha*, nebo *věk*. Ale jsou to pořád psi, takže sice mají **metodu** *Sedni, Lehni,* nebo *Spi*, ale nemají **metodu** *Létej,* která by se spíše měla vyskytovat ve třídě *TPtáci*.

Jak by vypadla taková třída se můžete podívat na Ukázce 3.1.1-1

Kromě toho, že objekty vycházejí ze tříd mají ještě další možnosti. Jednou z nich je také dědičnost. Představme si třídu *TAuto*. Je jasné, že tato třída má různé atributy, jaké by se dají u auta čekat (barva, max. rychlost, atd.). Ale co když chceme mít v programu objety jako *NakladakTatra*, ale i *VolkswagenBrouk*. Místo toho, abychom vytvářeli dvě různé třídy jako *TNakladniAuto* a *TOsobniAuto*, využijeme toho, že třída může být potomkem jiné třídy a tím pádem od ní převezme všechny její atributy a metody. Takže nejprve si vytvoříme třídu *TAuto*, která bude obsahovat všechny vlastnosti a metody, které by měly obě třídy společné, a následně se vytvoří další dvě třídy *TNakladniAuto* a *TOsobniAuto* odvozené do této třídy (viz. Ukázka 3.1.1-2).

```
TAuto = class // definice tridy
    Znacka: string;
    Typ: string;
    Barva: TColor;
    procedure Jed();
    procedure Zastav();
end;

TNakladniAuto = class(TAuto)
    VelikostKorby: integer;
    MaxNostost: real;
    procedure NalozKorbu();
    procedure VysypKorbu();
end;

TOsobniAuto = class(TAuto)
    PocetSedadel : integer;
    VelikostKufru: real;
    procedure NalozCloveka();
    procedure OtevriKufr();
end;
```

Ukázka 3.1.1-2 Ukázka zdrojového kódu návrhu třídy v programovacím jazyku Object Pascal, ukazující dědičnost tříd

Celá problematika OOP je samozřejmě mnohem složitější, ale účelem této práce není tuto problematiku vysvětlovat.

3.2 Vizuální programování

Při tvorbě aplikace obvykle nejdéle trvá návrh a vytvoření uživatelského rozhraní,

pro které se ujala zkratka GUI (Graphical User Interference). Vizuální programování spočívá především v tom, že se programátor GUI nevytváří psaním kódu, ale pomocí vývojového prostředí přidává na formulář (který představuje okno programu), komponenty, upravuje jejich vlastnosti, přidává jim potřebnou funkcionalitu a propojuje je mezi sebou. To vše s minimem psaní samotného kódu. Vývojové prostředí se potom samo postará o vykreslení jednotlivých komponent a nastavení jejich implementaci vlastností podle toho, co zadal programátor. Jak vypadá základ programu, který za nás vytvoří Lazarus se můžeme podívat v ukázka 3.2-1

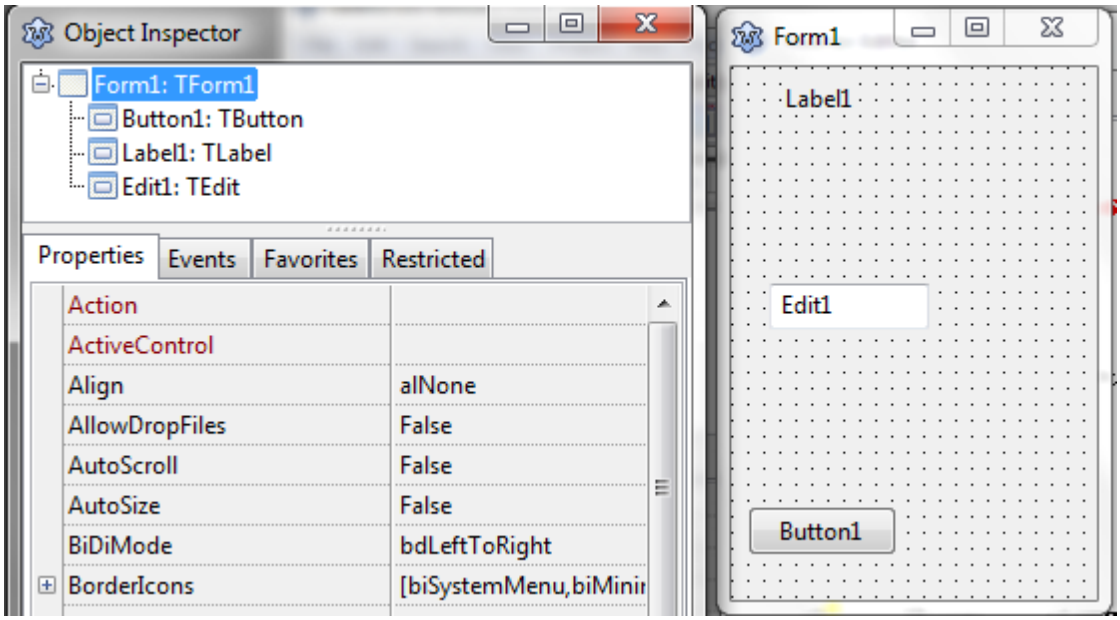
```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, LResources, Forms, Controls, Graphics,
Dialogs;

type
  TForm1 = class(TForm)
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  Form1: TForm1;

implementation

initialization
  {$I unit1.lrs}

end.
```

Ukázka 3.2-1 Takto vypadá zdrojový kód kostru programu, které Lazaus sám vygeneruje s každým novým projektem.

Lazarus pracuje s několika základními okny. Jsou to Form, Objekt Inspektor, Source Editor a Paleta komponent. Zatímco Source Editor slouží k samotnému psaní kódu, zbylá okna jsou zde především pro vytváření GUI.

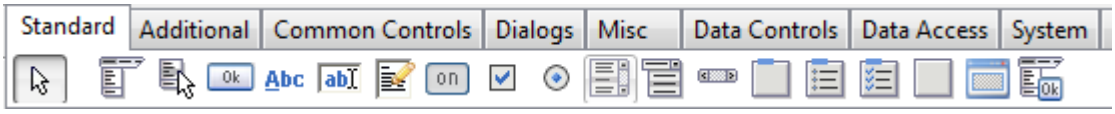
Okno Formulář (Obr. 3.2-1 vpravo) představuje v Lazaru okno budoucího programu. Jeho vlastnosti jako velikost, umístění, chování při minimalizaci atd. se dají

měnit pomocí dalšího okna: Objekt Inspektor (Obr. 3.2-1 vlevo). Na formulář se z Palety komponent dají přidávat další komponenty jako je Label, (popisek), Edit (editační pole), StringGrid (textová tabulka), Chart (graf) a jiné.



Obr. 3.2-1 Okno Formulář (vpravo) vývojového prostředí Lazarus, který představuje výsledné okno programu a okno Object Inspektor (vlevo), který slouží k úpravě jeho vlastností.

Paleta komponent (Obr. 3.2-2) je součástí hlavního okna Lazaru a právě na ní se dají najít jednotlivé komponenty. Pro větší přehlednosti jsou rozděleny do několika kategorií, jako je Standart, System, Advanced, Database atd... Všechny tyto komponenty se dají přidat na Formulář a poté nastavit v Objekt Inspektoru.



Obr. 3.2-2 Paleta komponent vývojového prostředí Lazarus.

Tímto procesem v podstatě pouze měníme třídu TForm, jejímž objektem je potom okno celého programu.

Každé okno programu je uloženo ve své vlastní *unitě*, která obsahuje zdrojový kód našeho formuláře, IDE ale potřebuje ještě další soubor, ve kterém budou uloženy

informace o formuláři (výška, šířka), i informace o každé další komponentě, kterou bude formulář obsahovat. V Delphi mají tyto soubory příponu *.dfm* v Lazaru pak *.lfm*.

Protože jeden program může obsahovat více oken (čili více formulářů), musí také existovat soubor, ve kterém je uvedeno, kolik a jakých formulářů náš program obsahuje. Tento soubor má v Delphi obvykle příponu *.dpr* a Lazaru potom příponu *.lpr*. Je to vlastně takový základní iniciátor celé aplikace. Když se na něj podíváte po tom, co ho pro vás Lazarus vytvořil, uvidíte kód uvedený v Ukázce 3.2-1

```
program Project1;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Interfaces, // this includes the LCL widgetset
  Forms, Unit1, LResources
  { you can add units after this };

{$IFDEF WINDOWS}{$R project1.rc}{$ENDIF}

begin
  {$I project1.lrs}
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

Ukázka 3.2-1 Zdrojový kód nového projektu v souboru lpr

Tam je vidět, že se nejprve příkazcem *Application.CreateForm(TForm1, Form1);* vytvoří formulář a následně se celá aplikace spustí příkazem *Application.Run;*

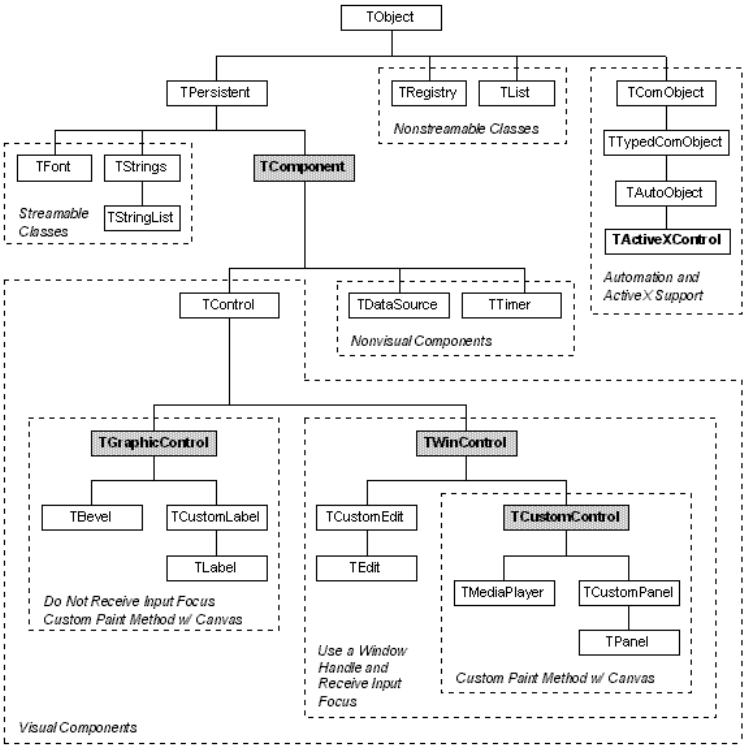
Největší pozitivum spatřujeme v tom, že tento soubor nemusíme během našeho programování vůbec zahlédnout, neboť Lazarus si ho (stejně jako mnoho jiných věcí) vytvoří a spravuje automaticky podle toho, co je právě potřeba.

Okna Source editoru, je poslední částí Lazarus o kterém jsme se chtěli zmínit. Právě tam dochází k editaci všech části zdrojového kódu programu. Tento editor je poměrně moderní. Je postaven na komponentě SynEdit, plně podporuje Unicode kódování cizích jazyků a obsahuje vlastnosti jako je **syntax highlight** (obarvení kódu, aby byl lépe rozeznatelný), **brace matching** (hledání související závorky), **auto**

indentation (automatické odsazení řádku), **auto completion** (automatické doplňování) nebo **code folding** (schopnost schovat kód patřící do jednoho logického bloku). Vše je pochopitelně plně nastavitelné. Sám Lazarus má mnoho klávesových zkratek, kterými se snaží ulehčit programátorovi práci. Vhledem k tomu, že průměrná unita s formulářem má okolo 150 řádků kódu, je to poměrně potřeba.

3.3 Komponenty

Jak jsme si již řekli, programování v Lazaru a Delphi stojí především na komponentách. Všechny jsou potomky třídy TObject a každá komponenta je potom instancí své třídy. Například komponenta Label je ve skutečnosti objektem třídy TLabel. Komponenty se dále dělí na vizuální a nevizuální. Vizuální jsou především ty, které se nějak vykreslují na formulář, nevizuální jsou například dialogy, jako je třeba dialog pro uložení souboru. Komponenty, které jsou dispozici hned po instalaci Lazaru, jsou součástí LCL, ale jak jsme si již řekli dříve, další se dají doinstalovat. Komponenty mají hierarchické uspořádání, což vychází z toho, že jsou to vlastně objekty.



Obrázek 3.3.1 Hierarchie komponent ve LCL [11]

Každá komponenta má následující:

Properities znamená česky vlastnosti a jsou to atributy daného objektu. Obvyklá je u vizuálních komponent především, výška, šířka, dále umístnění na formuláři, písmo, barva pozadí atd. Většina komponent má kromě vlastností, které jsou viditelné v Objekt Inspektoru i další, ale ty jsou přístupné pouze za běhu programu. Typickým případem je například informace o tom, které buňky v textové tabulce jsou momentálně označené. Některé vlastnosti jsou pouze pro čtení, takže je nelze za běhu programu měnit.

Methods jsou v češtině metody a jsou to metody daného objektu. Zjednodušeně lze říci, že jsou to funkce, které mohou být v programu zavolány a které souvisí s danou komponentou. Typickým případem je třeba vymazání editační políčka, nebo metoda, která zkopíruje obsah textového pole do schránky. Metody jsou již předem vytvořené a právě díky tomu, že většina komponent je vytvořena se sadou metod, které obsluhují velké množství základních úkonů, které bychom chtěli s danou komponentou provádět, může se programátor soustředit především na řešení hlavní funkcionality aplikace a nemusí se zabývat funkčností jednotlivých komponent.

Events znamená v českém jazyce události. Událost je metoda, kterou zavolá systém jako reakce na nějakou akci, vetšinou způsobenou uživatelem. Důležité je, že obsah této metody si můžeme napsat sami. Když například v programu poklepeme na tlačítko *Button*, zavolá se událost *TButton.OnClick.* Jak bude program na tuto událost reagovat závisí pak na nás, podle toho jaký kód do této události vložíme. Události jsou spolu s OOP základem vizuálního programování v Lazaru a Delphi.

3.3.1 Komponenta TStringGrid

První komponentu, na kterou se podíváme poněkud blíže je TStringGrid. Tato vizuální komponenta se nachází v kategorii Additional a slouží především k prezentaci a úpravě dat. Je typickou tabulkou, jakou můžete najít například v programu MS Excel nebo v OpenOffice Calk (Obr. 3.3.1-1).

#	Bmax [nmol]	Bmax [fmol/m.g.]	Bílkovina	Skupina
1	0,07	1,01	23,27	1
2	0,01	1	25,49	1
3	0,03	1,01	16,52	1
4	0,19	1,04	15,03	1
5	0,28	1,06	26,94	2
6	0	1	7,46	2
7	0,23	1,05	15,18	2

Obr. 3.3.1-1 Komponenta TSringGrid s vypsanými daty

Kromě obvyklých vlastností jako je výška, šířka, pozice atd. má především atributy *ColCount* a *RowCount*, které udávají a umožnují nastavit, kolik má tato tabulka sloupců, resp. řádků. Další důležité atributy jsou *FixedCol* a *FixedRow*, které udávají kolik sloupců, resp. řádků v tabulce bude zobrazeno jako fixní záhlaví, nebo jako fixní sloupce na levé straně tabulky.

Nejdůležitější atributem je potom *Cells*, což je v podstatě dvourozměrné pole textových řetězců o stejných rozměrech jako má tabulka, kdy každá položka pole odpovídá textu v každé buňce tabulky. Jednoduchým příkazem *StringGrid.Cells[0,0]:='Něco'* pak například dokážeme v programu přiřadit buňce úplně nahoře vlevo text „Něco“.

Dobrá vlastnost této komponenty je, že pokud máme hodně buněk, ale tabulka je omezená rozměry na formuláři, automaticky okolo sebe vytvoří posuvníky, jak je například vidět na Obr. 3.3.1-1.

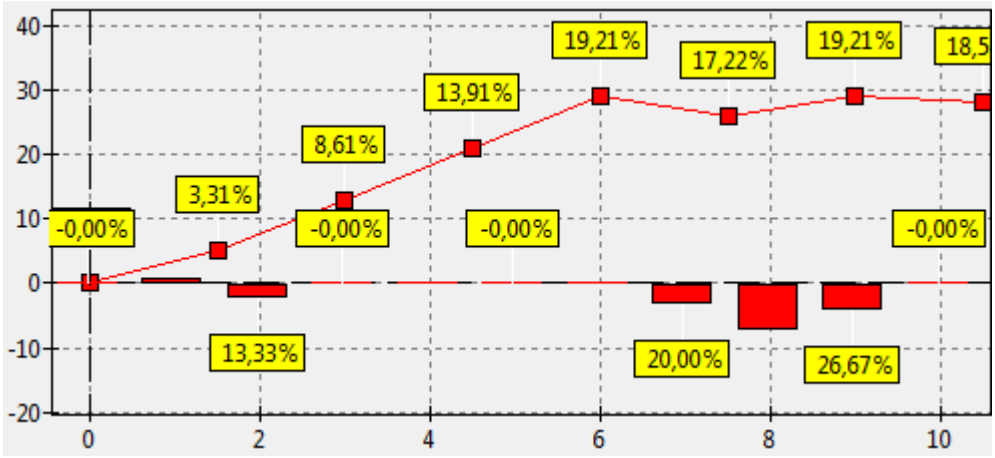
3.3.2 Komponenta TChart

Tato vizuální komponenta slouží v Lazaru k prezentaci obrazových dat. Může vykreslit klasický čárový, nebo sloupcový graf, i výsečový a plošný graf. Dokonce je možné v rámci jedné komponenty vykreslit najednou různé typy grafů.

Základem pro práci s daty v tomto grafu je třída TSeries, resp její potomci TBarSeries, TLineSeries, atd. podle toho jaký typ dat chceme vykreslit. Tato třída přestavuje sérii dat v grafu. Instanci této třídy je nejprve potřeba spojit s grafem metodou *TChart.AddSeries(TSeries)*. Nyní je tato série spojena s grafem a kdykoliv

něco v této sérii změníme, automaticky se tato změna promítne do komponenty grafu.

Každá třída obsahuje seznam bodů, které třeba u čárového grafu znamenají jednotlivé body křivky, u sloupcového jednotlivé sloupce atd. Tyto body lze přidávat pomocí metody *TSeries.Add*, a mazat pomocí metody *TSeries.Delete*. Další metody mají podobné názvy.



Obr. 3.3.2-1 Komponenta TChart, která má vykreslené dva druhy grafů.

Samozřejmě, že komponenta TChart má ještě mnoho dalších atributů a metod, díky kterým se dá nastavit skoro cokoliv od barvy popisek, přes možnosti zobrazení jednotlivých os, jejich popisky, jednotky, vzdálenosti, barvy a styly čar atd.

Další příjemnou vlastností této komponenty je, že při změně své velikosti automaticky roztáhne velikost grafu tak, aby prostor komponenty byl celý využit.

3.4 Problémy s vývojovým prostředím Lazarus

Nutno dodat, že celé prostředí je velmi dobře použitelné, což dokazuje mnoho projektů, které jsou v něm vytvořeny. Na druhou stranu jsem si při vývoji nemohl nevšimnut několika chyb, které mi vývoj poněkud znepříjemnily. Všechny tyto chyby jsou nahlášeny vývojářům Lazaru a je pravděpodobné, že budou v blízké době opraveny. Za dobu, kdy jsem s tímto vývojovým programem pracoval, se nová verze však neobjevila.

První závažnou chybou je špatné chování atributu *TForm.ModalResult*, která by měla obsahovat informaci o tom, jak bylo okno (formulář) zavřen. Bohužel, tento atribut není možno nijak donutit, aby vracel jinou hodnotu, než tu, která znamená, že okno bylo zavřeno křížkem v rohu.

Druhou chybou bylo chování komponenty *TButtonPanel*. Tato komponenta představuje panel s několika tlačítky, jako je OK, Cancel, Hepl atd. Bohužel, tato komponenta neprovede správně kód, který je možno doplnit do události *OnClick* každého tlačítka, resp. tento kód ignoruje a akci neprovede. To činí tuto, jinak velmi užitečnou komponentu, neupotřebitelnou.

Poslední chyba je spíše zvláštností. Výsledný exe soubor, ve kterém se nachází aplikace vytvořená v Lazaru, má obvykle o několik megabytu více, než by měla podobná aplikace vytvořená v Delphi. Je to z toho důvodu, že Lazarus přidává do výsledného exe souboru i ladící informace (neboli rutiny, které se starají o to co se stane, pokud se v programu vyskytne chyba). Toto chování se samozřejmé dá vypnout, avšak v dnešní době, kdy mají pevné disky stovky gigabytů, to není nutné.

4 Vazebné studie

4.1 Úvod do vazebných studií

Poslední dvě desetiletí byla svědky prudkého nárůstu našeho chápání existence velkého množství fyziologických receptorů a současně vývoje našeho porozumění základní strukturální motivaci a bio-chemickým mechanismům, které receptory provázení.

Právě vazebné studie se používají pro zjištění afinity různých druhů léků, drog či jiných látek na receptory nebo pro zjištění síly vazby stejných receptorových rodin, či jejich subtypů v různých tkáních nebo vzorcích. Těmto látkám, které se vážou na receptor, se obecně říká *ligandy*.

Výraz *receptor* se pracovně používá k popisu jakékoli buněčné makromolekuly, na kterou se látka váže, aby iniciovala své účinky. Funkce fyziologických receptorů zahrnuje vazbu příslušného ligandu a v následující odpovědi šíření jeho regulačního

signálu v cílové buňce.

Dejme tomu, že chceme zjistit jak se některá látka váže na stejný typ receptorů, které jsou umístěny různě v těle. Pokud bychom například chtěli cílit novou léčbu specifiky na srdce, tak by bylo výhodné najít takový ligand, jenž by se vázal pouze na receptory, které se vyskytují ve tkáních srdce.

Právě tyto studie nám pomáhají zjistit, zda lék bude mít nějaký terapeuticky příznivý nebo nepříznivý efekt na různé podtypy receptorů.

V této práci se budu zabývat dvěma základními metodami experimentálního zjišťování vlastností receptorů.

Saturační experiment je používán pro zjištění afinity mezi ligandem a receptorem a pro zjištění hodnoty maximální vazby mezi receptorem a ligandem, který se na receptor specificky váže.

Druhý experiment je takzvaná **OPA** (one point assay). V češtině se význam této zkratky dá popsat jako **jednobodová saturace**. Slouží pro porovnání dvou vzorků a zjištění, ve kterém došlo k navázání většího množství ligandu.

4.2 Základní pojetí vazebných studií

Základem ve vazebných studií je vazba ligandu (**L**) na receptor (**R**) a vytvoření takzvaného ligand-receptorového komplexu (**LR**). Tomu se také někdy může říkat vazba (podle anglického bound) a jedná se o množství ligandu, které se navázalo na receptor. To se dá označit jako **B**.

Nenavázaný ligand může být také označen jako **F** podle anglického slova *free*, neboť to označuje množství receptoru, které se nenavázalo a může reagovat s receptorem.

$$[L] + [R] \underset{k2}{\overset{k1}{\rightleftharpoons}} [LR]$$

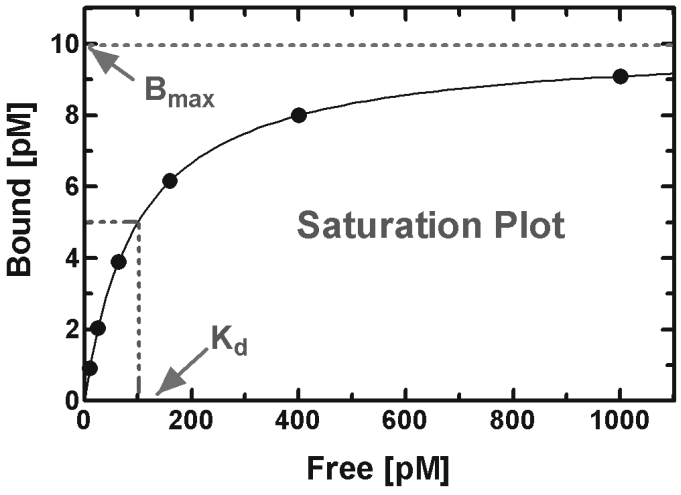
$$[R] + [F] \underset{k2}{\overset{k1}{\rightleftharpoons}} [B]$$

Rovnice 4.2-1 Kinetická rovnice vazby receptoru na lignad, kdy k1 a k2 jsou rychlostní konstanty asociační a disociační reakce. [12]

Měřeným parametrem je množství ligandu, které je navázané na receptor.

Důležitou konstantou je v tomto případě tzv. disociační konstanta, označovaná Kd. Ta se dá považovat za měřítko afinity ligandu k receptoru a dá se vyjádřit jako poměr k2/k1 (viz Rovnice 4.2.-). Tato konstanta také odpovídá koncentraci látky, při níž se obsadí 50% všech přítomných receptorů (viz Obr. 4.2-1). Čím je tato koncentrace nižší, tím je zřejmě vyšší afinita látky k receptorům.

Vysoká afinita látky k receptoru znamená nízkou disociační konstantu komplexu látky s receptorem. Tato konstanta charakterizuje příslušnou látku vzhledem k určitému receptoru, což je významné nejen pro klasifikaci látky, ale i pro predikci odpovědi na ni.



*Obr. 4.2-1 Standardní saturační křivka v vyznačenými body **Kd** a **Bmax** [12]*

Další důležitou hodnotou je Bmax. Ta vypovídá o hustotě receptorů v daném vzorku nebo tkáni a je ekvivalentní k bodu, kdy jsou všechny receptory obsazeny

ligandy a na saturační křivce (Obr. 4.2-1) odpovídá asymptotě, ke které se hyperbola blíží.

4.3 Základní postup v experimentu vazebných studií

Nejprve je třeba vzít vzorek tkáně, ve které chceme měření provádět. Receptory se nacházejí na membránách buněk, takže je třeba tyto buňky lyzovat a promýt, aby se ve vzorku nacházely především membrány s receptory.

Po tom, co se vzorky umístí do zkumavek, je do nich přidán radioligand. Tato látka je v podstatě normální ligand, který se specificky váže na receptor, až na to, že jeden z atomů jeho molekuly je nahrazen radioaktivním izotopem téhož prvku. Je známým faktem, že dva různé izotopy stejného prvku mají naprosto shodné chemické vlastnosti, takže výsledná molekula se bude v chemických reakcích chovat úplně stejně, jako by se v nich chovala molekula původního ligandu. Jak již bylo řečeno v kapitole 1.4, v mnoha případech se jako radioaktivní izotop často využívá ^{3}H, čili tricium.

Nyní se vytvoří dva druhy zkumavek. Do prvního se přidá vzorek tkáně spolu s radioligandem a do druhého se přidá ten samý vzorek a radioligand, ale také spolu s antagonistou. Antagonista je také ligand, ale má silnější vazbu na receptor, než má radioligand a navíc neobsahuje radioizotop.

Antagonista se přidává proto, že radioligand se může vázat i jinam než specificky na námi požadovaný receptor. Nyní máme dvě zkumavky, kdy v první se radioligand navázal nejen na receptor, ale také různě jinde. Tuto zkumavku můžeme označit jako celková aktivita (**C**). Ve druhé zkumavce nedošlo k navázání radioligandu na receptor, neboť ten je blokován antagonistou, ale radioligand se mohl navázat jinde na vzorek. Toto zkumavku můžeme označit jako nespecifická aktivita (**N**).

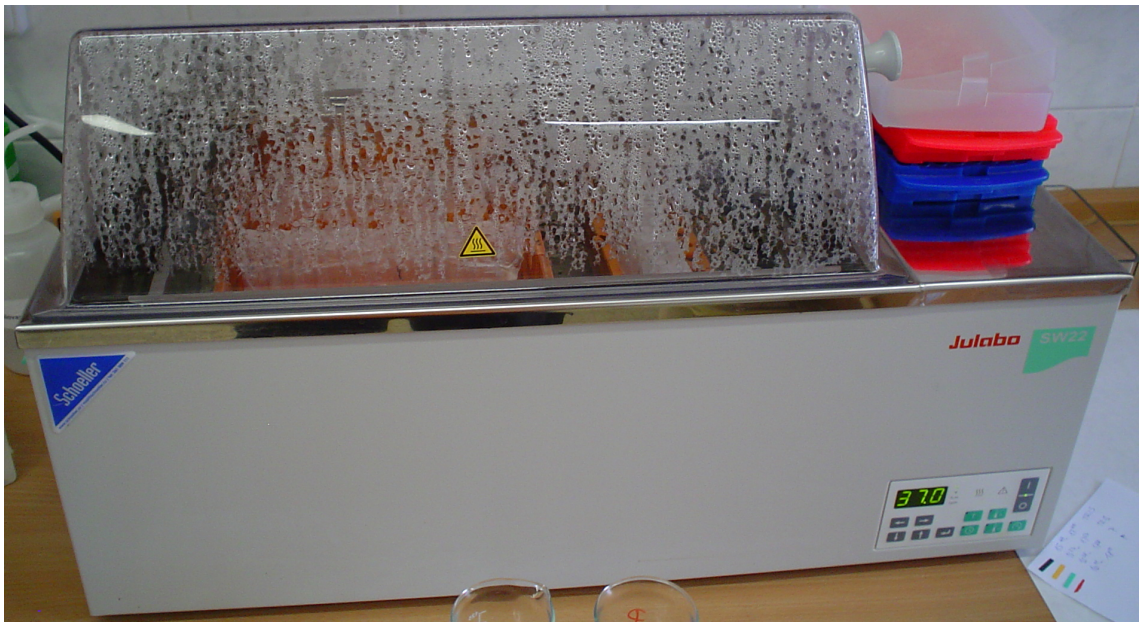
V pokusech, které momentálně probíhají ve fyziologické laboratoři se používá radioligand ^{3}H-1-azabicyclo[2.2.2]oct-3-yl hydroxy(diphenyl)acetate, neboli v angličtině 3H-quinclidinyl benzilate, se zkratkou ^{3}H-QNB.

Je běžné, že se vytvoří více zkumavek se stejnou koncentrací radioligandu a koncentrací vzorků tkáně, čili zkumavky jsou duplicitní, triplicitní nebo teoreticky

libovolně násobné. Čím vyšší násobek, tím vyšší je přesnost celého měření.

Kromě radioligandu, vzorku tkáně a antagonisty se ještě do každé zkumavky přidá pufr, který zajistí stálé prostředí pro vazebnou reakci.

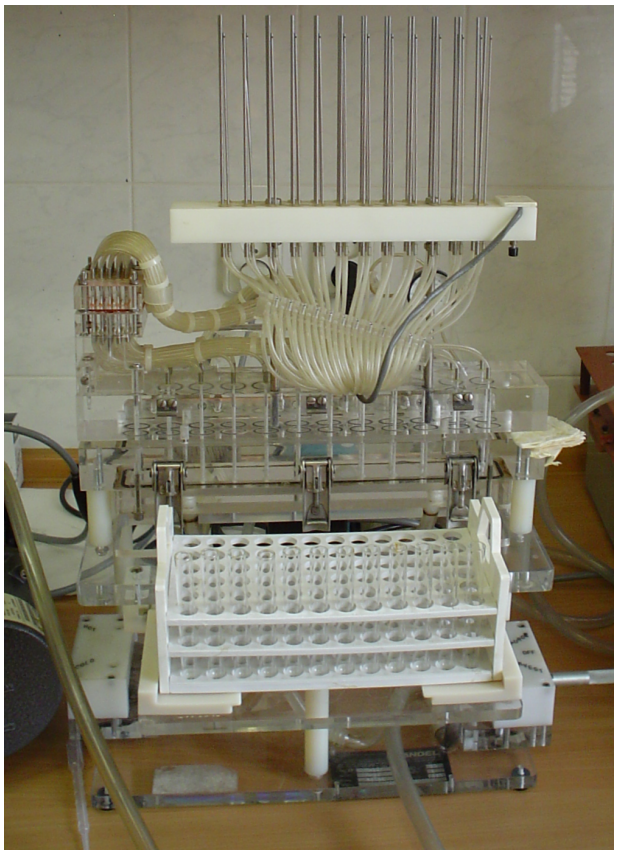
Nyní je třeba nechat zkumavky jistou dobu inkubovat. Teplota a čas inkubace závisí na vlastnostech ligandu a receptoru. V našem případě se používá inkubátor (Obr. 4.3-1) nastavený na 37°C, kde vzorky stráví 2 hodiny. Inkubace je nutná, aby se ve vzorku ustálila rovnováha mezi radioligandy a receptory podle Rovnice 4.2-1.



Obr. 4.3-1 Inkubátor ve kterém dochází za stále teploty k navázání ligandů na receptory

Další částí je filtrace, kterou ze zkumavek dostaneme volný radioligand. Je možno použít také centrifugaci, ale u obou je nutno dodržet co nejkratší časový interval, aby nedošlo k odloučení již navázaných ligandů od receptorů.

V našem případě se použila filtrace na speciální koloně (Obr. 4.3-2), kde jsou vzorky promývány skrze umělou membránu. Jelikož jsou receptory umístěny na buněčných membránách, neprojdou receptory s navázanými radioligandy přes filtr. Naopak nenavázané radioligandy projdou skrz filtr a jsou odvedeny pryč.



Obr. 4.3-2 Filtrační kolona, kde dochází k oddělení ligand-receptorových komplexů od nenavázaného ligandu.

Membrána z filtru se nechá vysušit (většinou 24 hodin) a následně se se vzorek z membrány přemístí do scintilačky (viz Obr. 1.4-3) spolu s kapalným scintilátorem (naftalen rozpuštěný v dioxanu). Jelikož radionuklid je již přítomen ve vzorku, je možno takto připravenou scintilačku vložit do scintilačního počítače.

Při použití radioaktivních ligandů je možné sledovat celkovou vazbu látky, ale současně i nespecifickou vazbu. Rozdíl mezi celkovou vazbou a nespecifickou vazbou umožňuje změřit specifickou receptorovou vazbu. Pokud od sebe odečteme aktivitu vzorků označených jako celková aktivita (**C**) a nespecifická aktivita (**N**) dostaneme tedy skutečnou aktivitu vzorku.

Abychom měli jistotu, že jsme pracovali správně je ještě dobré připravit si další zkumavku, ve které bude pouze čistý radioligand a kapalný scintilátor ve stejném množství jako bylo přidáno do ostatních scintilaček. Tomuto vzorku se říká pozitivní kontrola (**L**).

Jelikož zjišťovat koncentraci ligand-receptorových komplexů ve vzorku by nebylo dostatečně přesné, používá se přepočet na koncentraci bílkoviny ve vzorku. Proto je také potřeba zjistit koncentraci bílkovin v každém vzorku. To se provádí pomocí spektrofotometrie.

Jestliže tedy známe koncentraci bílkoviny ve vzorku a objem vzorku, můžeme již snadno přepočítat koncentraci ligand-receptorových komplexů na množství bílkoviny. Celková specifická radioaktivita ligandu vázaného na receptory, vztažená na mg tkáňové bílkoviny, je mírou hustoty receptorů.

Další důležitou hodnotou je specifická aktivita, označovaná zkratkou *As*. Tato veličina má jednotky DPM / fmol, čili ukazuje jaká je spojitost mezi aktivitou vzorků (rozpadů radioligandu za minutu) a koncentrací ligand-receptorových komplexů.

Hodnota As je specifická pro daný ligand a receptor. V laboratoři se obvykle nijak neměří, nýbrž její hodnotu dostane laboratoř spolu s dodávkou radioligandu. Ve firmách, kde se tyto radioligandy vyrábějí, se pro zjištění hodnoty specifické aktivity používá obvykle nějaký druh vysokotlaké kapalinové chromatografie (HPLC).

4.4 Saturace

V případě saturačního pokusu je úkolem vytvořit model saturační křivky (viz Obr. 4.2-1) a následně vypočtená data použít k výpočtu disociační konstanty.

V tomto případě se připraví jeden vzorek tkáně, který ve stejné koncentraci vložíme do několika zkumavek. Do každé z nich je přidána různá koncentrace radioligandu a tak budeme potřebovat tolik vzorků pozitivní kontroly, kolik různých koncentrací radioligandu jsme použili.

Tabulka 4.4-1 Různé koncentrace ligandu pro saturační experiment s 6 různými koncentracemi [12]

Zkumavky	Množství ligandu
1-4	0,1024 x Kd
5-8	0,256 x Kd
9-12	0,64 x Kd
13-16	1,6 x Kd
17-20	4 x Kd
21-24	10 x Kd

Při klasickém saturačním experimentu se používá 6 různých koncentrací radioligandu. Tři tyto koncentrace by měly být nižší než Kd a tři by měli být vyšší než Kd. Maximální hodnota koncentrace by měla být desetinásobek Kd, neboť při této koncentraci ligandu se téměř jistě pohybujeme na části saturační křivky, která odpovídá Bmax. Koncentrace ligandů lze najít v tabulce 4.4-1.

Pokud chceme vytvořit křivku s 6 body (což se obvykle dělá), za předpokladu, že budeme pracovat s duplicitními vzorky, budeme potřebovat 12 zkumavek s celkovou aktivitou, 12 zkumavek s nespecifickou aktivitou a 12 zkumavek s pozitivní kontrolou, které představují koncentrace ligandů. Postup při výpočtu vazby je uveden v rovnici 4.4-1.

$$B[\,fmol\,]=\frac{\bar{C}-\bar{N}}{As}$$

$$B[\,fmol\cdot mg\ prot^{-1}\,]=\frac{\bar{C}-\bar{N}}{As\cdot Prot\cdot V}$$

Rovnice 4.4-1 Rovnice popisující výpočet vazby v saturačním experimentu. ***As*** *je specifická aktivita,* ***Prot*** *je koncentrace bílkoviny a* ***V*** *objem vzorku.*

Pro každý bod nejprve uděláme průměr pro celkovou (**C**) a nespecifickou aktivitu (**N**) a tyto hodnoty od sebe odečteme. Získáme tak hodnoty aktivity vzorku, kterou když vydělíme specifickou aktivitou As, dostaneme koncentraci receptor-liganových komplexů, neboli vazby. Tato hodnota se ještě musí přepočítat úměrně na koncentraci bílkoviny vzorku, která představuje osu Y na grafu saturační křivky.

Výpočet nenavázaného ligandu (Free), který přestavuje osu X na grafu saturační křivky získáme jednoduchým odečtením celkové vazby od hodnot aktivity vzorku pozitivní kontroly. Tuto hodnotu vydělíme Specifickou vazbou (*As*) a můžeme jej přepočítat na hodnotu v závislosti na koncentraci na bílkovině. Viz Rovnice 4.4-2

$$F[\,fmol\,]=\frac{\bar{C}-\bar{L}}{As}$$
$$F[\,fmol\cdot mg\ prot^{-1}\,]=\frac{\bar{C}-\bar{L}}{As\cdot Prot\cdot V}$$

Rovnice 4.4-2 Rovnice popisující výpočet nenavázaného ligandu v saturačním experimentu. ***As*** *je specifická aktivita,* ***Prot*** *je koncentrace bílkoviny a* ***V*** *objem vzorku.*

Pro přesný výpočet saturační křivky je nutné použít nelineární regresi pro přesný výpočet hyperboly. Navíc je nutno použít software, který je certifikován pro laboratorní výpočty.

Aplikace Boreas tedy vytváří pouze přehledový graf a vypočítává data, které jsou posléze vyexportována a vložena do programu *GraphPad Prism verze 5.02*.

Exportovaná data jsou pro každý vzorek jeho hodnoty koncentrace volného ligandu (Free) a hodnoty celkové a nespecifické vazby. Vše v přepočtu na koncentraci bílkoviny.

Z tohoto experimentu dostaneme nejen hodnotu Kd, ale i další údaje, které mohou vypovídat o vlastnostech ligand-receptorových komplexů.

4.5 OPA

Jak již bylo dříve řečeno, OPA je zkratka pro One point assay, což se obvykle překládánno jako jednobodová saturace. Slouží pro porovnání dvou vzorků a zjištění, nebo porovnání dvou skupin vzorků.

Dejme tomu, že několika testovacím zvířatům dáme lék, který by měl ovlivňovat počet určitých receptorů. Nyní potřebujeme zjistit, jak onen lék fungoval. Z laboratorních zvířat vezmeme vzorky a zpracujeme je, postupem zmíněným v kapitole 4.3.

Je potřeba změřit koncentraci bílkoviny pro každý vzorek, protože používáme (relativně) rozdílné kusy tkáni.

Co se týče pozitivní kontroly, tak se nejčastěji používá 1 vzorek (přidáváme vždy stejné množství radioligandu) v duplicitním množství, ze kterého se ve výsledku vytvoří průměr.

Důležité pro tento pokus je zvolit takové množství radioligandu, abychom měli jistotu, že se pohybujeme v části saturační křivky, která se již blíží své asymptotě a tedy i hodnotě *Bmax*. Jen připomínám, že při této hodnotě by měly být všechny receptory obsazeny ligandem.

Můžeme koncentraci nastavit na hodnotu 10 krát vyšší než je hodnota *Kd*, abychom měli jistotu, že již jsme dosáhli plné saturace.

Kromě obvyklých hodnot jako je objem vzorku budeme potřebovat ještě hodnotu disociační konstanty *Kd*, abychom mohli vypočítat hodnotu *Bmax*.

$$F = \frac{\bar{L} \cdot As}{V}$$

$$B[\,fmol\,] = \frac{(\bar{C} - \bar{N})}{As} \cdot \frac{F \cdot Kd}{F}$$

$$B[\,fmol \cdot mg\ prot^{-1}\,] = \frac{\frac{A}{As} \cdot \frac{F \cdot Kd}{F}}{Prot}$$

*Rovnice 4.5.1 Výpočet hodnoty **B**, představující hodnotu koncentrace receptorů. **V** představuje objem vzorku a **Prot** je koncentrace bílkoviny*

Výpočet v tomto případě není složitý. Pro každý vzorek hledáme hodnotu *B*, která představuje koncentraci ligand-receptorových komplexů. V našem případě jsou všechny receptory obsazené ligandy, čili hodnota *B* přestavuje i hodnoty koncentrace receptorů.

Ve výsledku můžeme zprůměrovat hodnoty *B* pro vzorky v jedné skupině a uvidíme tak, jestli se dostavil efekt léku. Dále je možno s daty provést nějakou z metod statistické analýzy, abychom zjistili, jakou váhu mají naše naměřená data.

Statistické zpracování dat se opět provádí v programu *GraphPad Prism verze 5.02* a aplikace Boreas vypočítává hodnoty *B*.

IV. Praktická část

5 Vytvoření spojení mezi osobním počítačem a scintilačním počítačem po sériovém portu.

Tato část začala teoretickou přípravou, kdy byli zjišťovány podrobnosti ohledně komunikace sériového portu.

Bylo nutné mít přehled o možnostech konfigurace spojení kvůli navštívení fyziologického ústavu, aby bylo možno provést co možná největšího množství pokusů když bude k dispozici scintilační počítač. Manuál ke scintilačnímu počítači *Beckman LS 1801* se nacházel na půdě fyziologického ústavu v jednom exempláři. Je poměrně obsáhlý a nebyla možnost číst ho před návštěvou na výše zmíněném pracovišti.

Jelikož testovací notebook nebyl vybaven sériovým portem a o počítačovém vybavení fyziologického ústavu byli jen velmi neodborné informace, bylo třeba najít nějakou alternativu pro připojení na sériový port.

Z poměrně širokého spektra různých přídavných karet a podobných zařízení pro přidání sériového portu do počítače, byl nakonec vybrán *PremiumCord převodník z USB2.0 na RS232 s kabelem*. Tento produkt sliboval vysokou mírů funkčnosti, je vybaven čipem FT8U2xxAM, konkrétně **FT8U232AM** a přenosovou rychlost až do 500Kbps. Recenze ohledně tohoto produktu byly vesměs velmi pozitivní.

Další částí bylo softwarové vybavení. Pro prvotní navázání komunikace byl připraven program *Hercules SETUP utility* od *HW-group.cz*. Tato aplikace je univerzální utilita, která supluje terminál na sériovém portu (*RS-232* nebo *RS-485*) a *UDP/IP*, *TCP/IP* (klient nebo server) a je to užitečná pomůcka pro práci s Ethernet zařízeními. Dále byla vytvořena jednoduchá aplikace v Lazaru, která vypisovala obsah který se objevil na sériovém portu. Využívala knihovnu *Synaser* a měla ověřit, zda je možné na této knihovně postavit budoucí aplikaci.

Na ústavu byl vyhledán manuál a byli zjištěny parametry připojení. Následně se povedlo připojit oba programy na sériový port a získat data ze scintilačního počítače.

Nastavení sériového portu počítače bylo následující: 300 baud, 1 start bit, 8 data bitů a nulová parita (nebo 1 start bit, 7 data bitů + 1 paritní bit). Stop bity mohou být 2 nebo jen jeden.

Co se týče handshakingu, Beckman podporuje jak RTS/CTS tak i DSR/DTR handshaking. Vše musí být zapojené příslušné vodiče, což nás nemusí v případě převodníku zajímat neboť ten má zapojeny všechny vodiče a závisí pouze na nastavení portu. Samotný Beckam hardwarový hangshaking sice podporuje, ale nevyžaduje. Nejspíš by mu stačil i obyčejný třívodičový kabel (TxD, RxD a Ground).

Co se týče softwarového handshakingu, tam je situace poněkud složitější. Podle manuálu, Beckman tento typ podporuje a správně odpovídá na X-OFF signál, ale měl by fungovat i bez něj. Skutečnost je však poněkud jiná. V praxi se ukázalo, že pokud se rozhodnete nepoužívat tuto funkcionalitu, počítač se chová nepředvidatelně a je lepší ji nechat zapnutou, hlavně v případě, že je vypnut i hardwarový handshaking.

Část manuálu, která pojednává o komunikaci po sériovém portu, byla okopírována a na jejím základě byl řízen další vývoj aplikace. Tuto část manuálu je možno najít jako přílohu č.2.

6 Možné způsoby ovládání a komunikace se scintilačním počítačem

Analýzou manuálu a výstupu ze sériového portu se podařilo zjistit byla následující: Beckman komunikuje jen velmi jednoduše. Je potřeba, aby byla komunikace po sériovém portu zapnuta v jeho nastavení, které se musí provést po každém zapnutí přístroje.

Po sériovém portu dají data pouze číst. Jakýkoliv pokus o poslání dat do přístroje vyústí v chybu a nutnost restartovat Beckmana. V principu Beckman vypisuje na port téměř stejná data, která vypisuje na tiskárnu. Informace na portu jsou tedy standardní ASCII znaky. Pro porovnání se můžete podívat na přílohy č.3 a 4. Příloha číslo tři je záznam z komunikace po sériovém portu, jak jsem ho zaznamenal pomocí mého programu. Příloha č.4 je oskenovaná část tištěného výstupu z tiskárny.

Beckman vysílá data v datových blocích, kdy je každý datový blok označen číslem, které řídí druh datového bloku. Například začátek komunikace je iniciován blokem číslo 200 a obsahem tohoto bloku je datum, nastavení počítače atd.

Pro nás jsou důležité bloky s číslem **100** a **105**, které obsahují informace o hodnotě **CPM** a **DPM**. Tyto bloky obsahují ještě další informace jako je číslo vzorku, číslo kanálu, čas měření, číslo H a jiné.

Blok s číslem **100** obsahuje informace o číslu vzorku, číslo kanálu, CPM, času měření, a průměrné hodnotě H. Další dva údaje - DPM a efektivita jsou poslány hned v dalším bloku, tentokrát však s číslem **105**.

Jednotlivé hodnoty jsou od sebe oddělené čárkou. Desetinná čísla používají pro oddělení desetinné části tečku. Většina hodnot ale není pro naši aplikaci důležitá a jediné, co se z výpisu sbírá jsou hodnoty **DPM**.

Další důležitý blok má číslo **999**. Tento blok označuje, že scintilační počítač již nemá v zásobníku žádné další zkumavky a ukončuje měření.

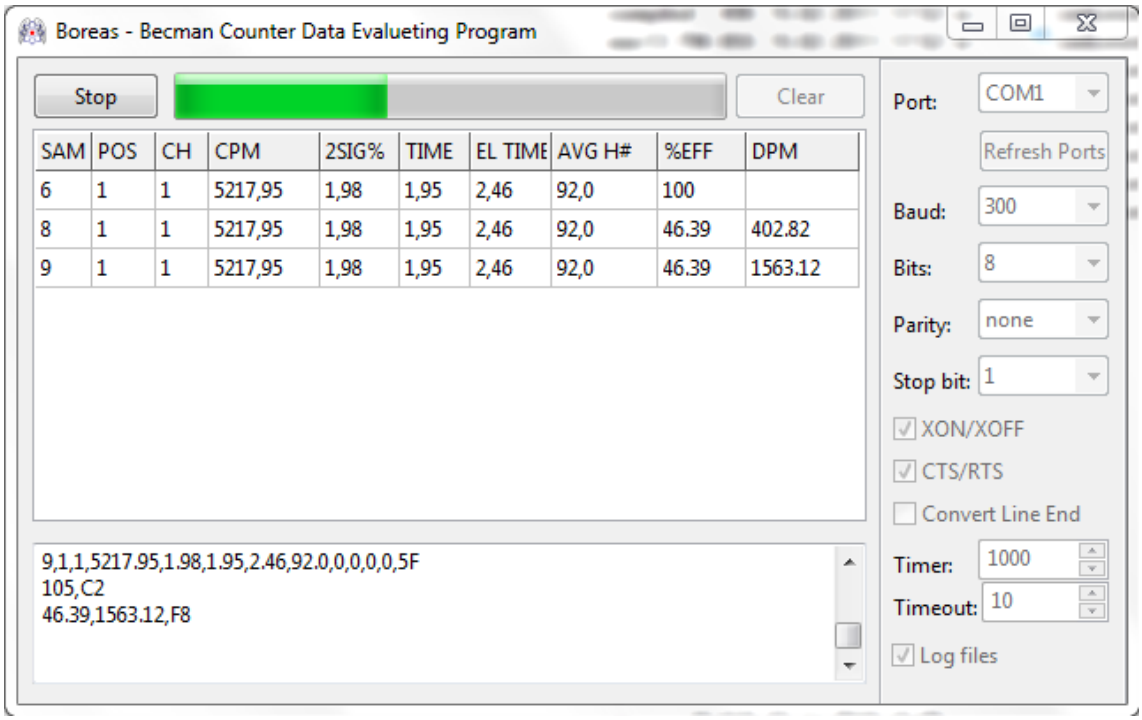
Na konci každého řádku se vždy vyskytují také 2 znaky, které slouží jako jednoduchý kontrolní součet obsahu bloku. V podstatě jde pouze o součet všech ASCII hodnot jednotlivých znaků na řádku a vyjádření této hodnoty v hexadecimální soustavě. Více informací můžete najít v příloze č.2.

Tento kontrolní součet není tak důležitý, uvážíme-li ostatní kontrolní mechanizmy přenosu, jako je handshaking sériového portu. Z toho důvodu nebyl kontrolní součet v nižších verzích aplikace Boreas nijak kontrolován. Až ve finální verzi byla vytvořena funkce, která sloužila k ověření pravosti kontrolního součtu.

Jelikož typický výpis ze sériového portu obsahuje některé netisknutelné znaky (*STX*, *ETX*) a nadbytečné mezery, je dobré tyto znaky před dalším zpracováním vypustit.

7 Vytvoření prototypu aplikace, určeného pouze ke sběru dat, které budou použity při tvorbě finální verze aplikace

První verze aplikace Boreas (viz Obr. 7-1) byla velmi jednoduchá. Skládala se především z jednoduchého panelu s nastavením parametrů a tabulky do které se vypisovali zpracované hodnoty.



Obr. 7-1 První verze aplikace Boreas

Program jako takový byl vytvořen v květnu, v roce 2010. Již tato jednoduchá aplikace splňovala polovinu zadaného úkolu. Dokázala komunikovat se scintilačním počítačem a sbírat hodnoty, které posílal na sériový port. Tudíž již nebylo nutné přepisovat data z papíru do počítače a stačilo potřebný sloupec (**DPM**) zkopírovat a vložit tam, kde byl potřeba.

Navíc aplikace sbírala a především uchovávala veškerou komunikaci s počítačem. To bylo důležité především pro analýzu komunikace a aby bylo možné zachytit datové bloky, které se vyskytovaly například při chybě nebo při nestandardním chování. Bez znalostí identifikačních kódů těchto bloků by nebylo možno ošetřit tyto výjimečné stavy v budoucí verzi aplikace Boreas.

Pro komunikaci po sériovém portu tato verze aplikace používala již dříve zmíněnou knihovnu *Synaser*. Její hlavní částí je třída *TBlockSerial*, a především její metoda *Recvstring*. Ta vrací textový řetězec obsahující jeden řádek, který byl odeslán na sériový port.

Opakovaným voláním této metody je potom přečten celý obsah komunikace. Bloky **100** a **105** jsou zpracovány a jejich obsah je po přeformátování zapsán do tabulky.

Pokud pomineme fakt, že tato aplikace nedokázala data vyhodnocovat, největší nevýhodou této verze aplikace bylo, že si neukládala žádné nastavení, a po každém zapnutí aplikace se museli znovu nastavovat parametry sériového portu.

Dále u této aplikace nebylo nijak vyřešené ukládání a opětovné prohlížení dat. Navíc program sbíral data, které nebyla pro další pokus potřeba a jen zabírala místo.

8 Návrh a tvorba finální verze aplikace vhodná pro laboratorní využití scintilačním počítače

8.1 Úvod

Finální verze aplikace Boreas byla od začátku plánována jako mnohem komplexnější aplikace, která bude vhodnější pro dlouhodobé používání.

První části návrhu aplikace bylo shromáždění požadavků od lidí, kteří tento software budou používat. Ve výsledku tedy měla být Boreas aplikace, která umožní načtení dat ze scintilačního počítače. Tyto data zobrazí v přehledné formě, vypočítá potřebné údaje a umožní jejich export ve zvoleném formátu. Také bylo nutné, aby umožňovala nastavit vlastnosti měření, ukládala si svoje nastavení a také aby umožňovala ukládání hodnot měření a jejich pozdější prohlížení.

Kvůli problémům s exportem dat bylo potřeba, aby existovala možnost nastavit formát, v jakém budou data vypisována (oddělovač desetinné části) a to nezávisle na nastavení sytému, na kterém program běží.

V rámci přehlednosti byl do návrhu aplikace přidán také graf, neboť vizuální

zobrazení hodnot pomáhá lepší orientaci v sebraných datech.

Protože hlavním úkolem aplikace bylo zpracovávat data a provádět výpočty, bylo nutné přesně zjistit hodnoty, které budou do programu vstupovat a jaké z něj budou vystupovat, včetně použitých jednotek. Poté byly sestaveny rovnice, které tyto přepočty prováděli a ty byly následně přepracovány do podoby počítačového algoritmu.

Tyto výpočetní rovnice byly ověřeny na jednom dříve proběhlém měření, kdy jsme se ujistili, že podávají stejné výsledky jako dosavadní výpočetní mechanizmus, který byl používán v laboratoři.

8.2 Uživatelské rozhraní

V dalším kroku byl vytvořen návrh uživatelského rozhraní aplikace a ten byl konzultován s budoucím uživatelem a vyladěn podle jeho představ, aby odpovídal reálným potřebám.

Okno aplikace Boreas (Obr. 8.2-1) má několik částí. Kromě hlavního menu, je to panel *Vlastnosti měření*, panel hodnot *DPM*, *Graf* a také *Tabulka výstupních dat*.

Velikost jednotlivých částí se dá jednoduše měnit pomocí posuvníků, kdy stačí stisknutím a tažením zvětšit či zmenšit rozměr panelů.

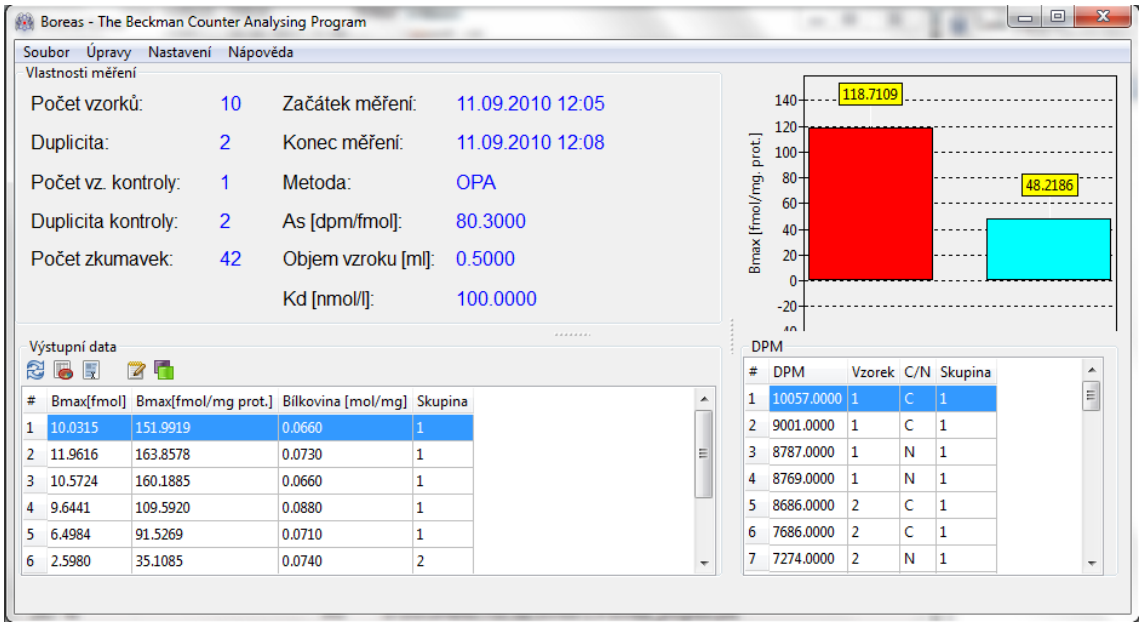
Pomocí hlavního menu můžeme ovládat většinu důležitých funkcí programu. Kromě základnách funkcí v částech **Soubor** a **Nápověda** jsou důležité zejména části **Úpravy** a **Nastavení**.

Panel *Vlastnosti měření* je pouze informační a slouží k zobrazení zadaných vlastností, jako je počet vzorků nebo jejich duplicita. Vlastnosti jako je datum začátku, datum konce měření a použitá metoda, nelze ovlivnit, ostatní se dají změnit pomocí *dialogu Vlastnosti měření*.

V tabulce výstupních dat aplikace zobrazuje data, která program vypočítal a která jsou určená pro export do další aplikace podle uvážení uživatele. Také se zde nachází tlačítko pro export dat.

V aplikaci Boreas graf slouží pouze jako přehled vypočtených hodnost pro jednotlivé metody měření.

Panel DPM obsahuje tabulku, která zobrazuje hodnoty jednotlivých zkumavek v **DPM**. U každé hodnoty je také uvedeno, kterému vzorku tato hodnota DPM patří a taká zda se jedná o údaj o **celkové aktivitě (C)** nebo údaj o **nespecifické aktivitě (N)**.



Obr. 8.2-1 Finální verze aplikace Boreas

Ikony použité v aplikaci Boreas, pochází z balíku *Tango Desktop Project.* Jsou zveřejněny pod licencí *Public Domain* a tudíž jsou svobodně k použití.

8.3 Třída TData

Hlavní částí aplikace je třída *TData*, která zapouzdřuje seznam **DPM** hodnot a další důležité veličiny, jako je specifická aktivita, počet vzorků, počet vzorků pozitivní kontroly, duplicita a jiné ve formě svých atributů.

V aplikaci se vyskytuje jeden objekt **Data**, který je instancí této třídy a spolu s definicí třídy je uložen v unitě *DataClass.pas*.

Seznam atributů třídy *TData* je možno najít v Ukázce 8.3-1, kde je vidět jaký datový typ se používá pro jednotlivé hodnoty.

Seznam hodnot **DPM** je uložen jako *TStringList*, čili jako seznam řetězců. To dovoluje jednoduchou editaci a úpravu těchto hodnot. Podobně je uložen i seznam jednotlivých hodnot bílkovin a skupin u OPA typu měření. Ostatní hodnoty jsou uloženy jako celočíselný datový typ (počet vzorků), nebo jako číselný typ s plovoucí čárkou (specifická aktivita), případně jako typ pro práci s datem a časem (*TDateTime*).

Tato třída také obsahuje nezbytné metody pro práci s daty. Ukládání a načítání dat ze souboru a exportu dat se věnuje **kapitola 8.5** a jednotlivým výpočtům dat **kapitola 8.6**.

```
TData = class
    Dpm: TStringList;         // seznam hodnot DPM
    Ref: TStringList;         // seznam hodnot bílkovin
    Group: TStringList;       // seznam skupin

    Method: TMethod;          // metoda měření
    Sam: word;                // počet vzorků
    Sd: byte;                 // duplicita vzorků
    Ssam: word;               // počet vzorků pozitivní kontroly
    Ssd: byte;                // duplicita vzorků pozitivní kontroly

    Ass: real;                // specifická aktivita
    Vol: real;                // objem vzorku
    Kd: real;                 // disociační konstanta
    Prot: real;               // koncentrace bílkoviny

    Delimeter: char;          // oddělovač desetinné části
    Digits: byte;             // počet číslic

    DateStart:TDateTime;      // datum a čas začátku měření
    DateEnd: TDateTime;       // datum a čas konce měření
```

Ukázka 8.3-1 Seznam atributů třídy TData.

8.4 Třída TSetting

Jelikož se bude aplikace reálně používat bylo záhodno, aby byla schopná pamatovat si jednotlivá nastavení, ať již se jedná o oddělovač desetinné čárky, nastavení sériového portu, či předvoleb jednotlivých metod měření.

Protože se většina experimentů s vazebnými studiemi bude provádět v několika sériích, bylo záhodno zajistit, aby při zadávání vlastností nového měření byli již předvyplněné nejčastější hodnoty a aby si tyto hodnoty nastavit uživatel dopředu.

Pro tento účel obsahuje aplikace třídu *TSetting*, která tyto nastavení zapouzdřuje a objekt **Setting**, obojí v unitě *SettingClass.pas*. Metoda *TSetting* zároveň obsahuje metody pro načítání a ukládání těchto nastavení do souboru.

Všechny tyto nastavení se ukládají do souboru **config.ini**, který se nachází ve stejném adresáři jako spustitelný soubor aplikace. Pokud tento soubor neexistuje, aplikace si ho sama vytvoří. Jelikož se jedná o klasický ini soubor, jehož struktura je poměrně přesně daná, lze se pro práci s ním využít třídu *TINIFile*, která práci s tímto formátem souboru značně ulehčuje a je součástí LCL.

Náhled všech atributů a metod třídy *TSetting* můžete najít v Ukázce 8.4-1.

V aplikaci lze většinu hodnot nastavení zadat pomocí dialogu **Předvolby**. Tento dialog má tři části, kdy první se věnuje obecným předvolbám aplikace, jako je zobrazovaný počet desetinných míst, znaky pro oddělovače desetinných míst a znaků pro export hodnot. Další dvě záložky jsou věnovány předvolbám pro měření saturace a OPA.

Nastavení sériového portu je dá upravit pomocí separátního dialogu **Sériový port**, které se dá vyvolat z části **Nastavení** z hlavní nabídky.

```
TSetting = class
  private
    function Nacti:boolean; // metoda pro načtení nastavení ze souboru
    ConfigFilePath: string; // adresář aplikace
  public
    constructor Init(const Dir:string); // konstruktor aplikace
    function Uloz:boolean; // metoda pro uložení nastavení ze souboru
    Delimeter: char;       // oddělovač desetinné části čísla
    Digits: byte;          // počet zobrazených desetinných čísel
    Eol: char;             // oddělovač při exportu
    Port: TPortSetting;    // nastavení sériového portu
    Opa: TOpaSetting;      // předvolba měření OPA
    Sat: TSatSetting;      // předvolba měření saturace
  end;
```

8.5 Vstupy a výstupy

Jelikož jednou z funkcí aplikace Boreas je ukládání a opětovné prohlížení jednotlivých měření, bylo nutné vytvořit metody pro ukládání a načítání hodnot ze souborů na pevném disku. Prvotní návrh počítal s ukládáním do databáze, ale nakonec

byl uživatelem zvolen způsob, kdy každé měření bude mít samostatný soubor.

Jelikož jednotlivé měření mohou obsahovat různý počet vzorků, není jasné jaký typ metody bude použit a je potřeba uchovávat rozdílné typy hodnot, byl pro ukládání zvolen formát **XML**.

Každý soubor je v podstatě textový dokument, formátovaný pomocí speciálních hodnot v textu. Prvních několik hodnot je shodných pro oba typy měření a další postup při načítání a ukládání je rozdílný a řídí se právě typem metody použité při měření.

Výhody **XML** formátu jsou, že tento formát je zcela otevřený a soubory s uloženými hodnotami se dají teoreticky otevřít v libovolné aplikaci, které dokáže s **XML** soubory pracovat.

IDE Lazarus navíc poskytuje velmi užitečné nástroje pro práci s tímto typem souborů jako je třída *TXMLDocument* a její metody *ReadXMLFile* a *CreateElement*.

Ve výsledku se o tyto operace starají metody třídy *TData* a to konkrétně *NactiData* a *UlozData*.

Jelikož aplikace bude potřebovat komunikovat s dalšími aplikacemi a předávat jim data, která sama zpracuje, obsahuje Boreas funkce pro export vypočtených dat do formátu **CSV**.

Teto formát je v podstatě jen textový soubor, kdy jsou jednotlivé hodnoty od sebe odděleny nějakým znakem. Problémem je, že různé typy programů využívají různé znaky, především z důvodu národnostních zvyklostí. Zkratka **CSV** znamená „*comma separated values*“, čili hodnoty oddělené čárkou. Jenže například v České republice se čárka používá pro oddělení desetinné části čísla a tudíž se v **CSV** místo čárky používá středník.

Proto aplikace Boreas umožňuje nastavení těchto parametrů nezávisle na nastavení systému.

Pro export do souboru **CSV** byla vytvořena funkce *GridToCSV*, která vezme data z výstupní tabulky a přeformátuje je podle nastavení programu. Nakonec je uloží jako text

do souboru, který zvolí uživatel.

O nové hodnoty DPM se stará formulář *Nové měření*. Před jeho spuštěním je uživatel dotázán na počet vzorků, duplicitu, hodnotu specifické aktivity a další potřebné veličiny. Formulář samotný potom čeká na připojení scintilačního počítače a automaticky sbírá potřebné hodnoty, podobně jako první verze aplikace Boreas.

Pro komunikaci po sériovém portu již není využívána knihovna *Synaser*, ale komponenta *TSdpoSerial*, která z této knihovny vychází. Tato komponenta má událost *OnRxData*, která se provede vždy, když na sériový port dorazí data. Tato událost data zpracovává a předává další části aplikace, podle toho o jaký blok dat se jedná. Viz **kapitola 6**.

8.6 Výpočtová část

Třída *TData* má metody pro práci s daty, kdy vedle metod pro ukládání a načítání hodnot z uloženého souboru, obsahuje především metody pro výpočet, výpis a vykreslení hodnot do grafu.

Pro tyto výpočty byly použity rovnice z **kapitoly 4** a algoritmy z **kapitoly 8.1**. Hlavní část výpočtu se prování v metodě třídy *TData.VypoctiTabulku*. Tato metoda na základě atributů třídy *TData* jako je počet vzorků, duplicita a metoda experimentu. Program na základě těchto dat sestaví potřebné rovnice a provede výpočty. Zdrojový kód metody *VypoctiTabulku* lze najít v příloze č.5.

Vypočtená data se exportují do tabulky komponenty *TStringGrid*. U metody OPA se pro každý vzorek vypisuje hodnota *B*, dále hodnota *B* přepočtená na mg proteinu, koncentrace bílkoviny a číslo skupiny, do které vzorek patří.

U saturačního experimentu se vypisují jednotlivé hodnoty *B* a *F* pro každý bod křivky, přepočtené na mg proteinu.

Jelikož aplikace obsahuje i graf, je třeba vypočítat i data pro grafy. Opět se vychází z rovnic z **kapitoly 4** a o výpočet a vykreslení se stará metoda *VypoctiGraf*, jejíž zdrojový kód můžete najít v příloze č.6.

Pro saturační experiment se vypočítají jednotlivé body saturační křivky a z těchto bodů se vytvoří čárový graf. Na ose **Y** se nachází hodnoty *B* a na ose **X** potom hodnoty *F*.

U metody OPA se nejprve vypočítají hodnoty *B* pro každý vzorek. Následně se vzorky roztřídí do skupin a vypočte se průměr jednotlivých skupin. Průměrné *B* pro každou skupinu se potom vykreslí do grafu jako jeden sloupec sloupcového grafu, kdy každá skupina je označena jinou barvou.

8.7 Používání aplikace

Pokud pracovník chce použít aplikaci, bude postupovat následovně: Nejprve si připraví vzorky a vloží je do scintilačního počítače. Následně zapne osobní počítač a spustí program Boreas. V menu zvolí položku nové měření podle měřící metody. Následně by vyplní potřebné údaje (ale většinou jsou již předvyplněné). V případě, že se jedná o měření OPA, zobrazí se ještě dialog pro hodnoty bílkovin a skupin.

Následně se zobrazí dialog *Načítání vzorků*, z kterého je jasné, který vzorek se momentálně načítá a kolik jich také zbývá do konce. Zobrazen je také čas, který aplikace odhaduje do konce měření. V tento moment je nutno zapnout scintilační počítač a pokud je vše správně nastaveno, není třeba se již do konce měření o nic starat.

Poté, co je odeslán dostatečný počet vzorků, se dialog automaticky uzavře a z naměřených hodnot se vypočítají potřebné výstupní hodnoty. Pak již stačí jen měření uložit a výsledky exportovat pro další zpracování.

Pro budoucí uživatele byla sepsána uživatelská nápověda k programu, která má zajistit správné používání a která je přístupná z hlavní nabídky programu.

Aplikaci se již testovala na fyziologickém ústavu a je připravená k nasazení do provozu, ale nejprve je třeba zaučit personál a připravit osobní počítač.

V. Závěr

Bakalářská práce splnila cíle, které si stanovila, a celý projekt tvorby aplikace pro sběr a vyhodnocení dat ze scintilačního počítače dospěl do finální fáze.

Jednotlivá měření nyní mohou probíhat mnohem rychleji a efektivněji než v minulosti a to díky tomu, že většinu práce se sběrem a výpočty se přesunula ze člověka na počítačový program. Tím se i snížilo množství chyb v tomto procesu.

Aplikaci budoucí uživatelé ocenili kladně, zejména její jednoduché a přehledné ovládání a také i její intuitivní grafické rozhraní.

Vývoj aplikace Boreas by ale touto bakalářskou prací neměl skončit. Pro další změny v aplikaci je ale třeba počkat na reakce uživatelů, kteří budou Boreas delší dobu používat. Nelze vyloučit ani změny v grafickém rozhraní.

Do budoucna je také například na fyziologickém ústavu naplánováno nahradit dosavadní jehličkovou tiskárnu, připojenou ke scintilačnímu počítači řádkovým displejem. Tiskárna je totiž díky aplikaci Boreas potřeba jen pro nastavení scintilačního počítače a na tento úkol by se více hodil řádkový displej připojený na tiskový port. Na tomto úkolu se již pracuje a tento přípravek by měl být již brzy k dispozici.

VI. Použitá literatura

[1] NAVRÁTIL, Leoš ; ROSINA, Jozef . *Medicínská biofyzika*. Praha : Grada, 2005. 524 s. ISBN -10:80-247-1152-4.

[2] ULLMANN, Vojtěch. *Astro Nukl Fyzika* [online]. 2005 [cit. 2011-04-24]. Detekce a spektrometrie ionizujícího záření. Dostupné z WWW: <http://www.astronuklfyzika.cz/DetekceSpektrometrie.htm>.

[3] VYKOPAL, Jan . *Builder* [online]. 2001 [cit. 2011-04-24]. Sériové rozhraní v Delphi. Dostupné z WWW: <http://www.builder.cz/art/delphi/delphiser.html>.

[4] OLMR, Vít. *HW.cz* [online]. 2005 [cit. 2011-04-24]. HW server představuje - Sériová linka RS-232. Dostupné z WWW: <http://hw.cz/rs-232>

[5] TIŠNOVSKÝ, Pavel. *Root.cz* [online]. 2008 [cit. 2011-04-24]. Sériový port RS-232C. Dostupné z WWW: <http://www.root.cz/clanky/seriovy-port-rs-232c/>.

[6] TIŠNOVSKÝ, Pavel. *Root.cz* [online]. 2008 [cit. 2011-04-24]. Komunikace pomocí sériového portu RS-232C. Dostupné z WWW: <http://www.root.cz/clanky/komunikace-pomoci-serioveho-portu-rs-232c/>.

[7] TIŠNOVSKÝ, Pavel. *Root.cz* [online]. 2008 [cit. 2011-04-24]. Komunikace pomocí sériového portu RS-232C podruhé. Dostupné z WWW: <http://www.root.cz/clanky/komunikace-pomoci-serioveho-portu-rs-232c-podruhe/>.

[8] POLÁCH, Eduard. *Programování v jazyku Turbo Pascal*. České Budějovice : Pedagogická fakulta JU, 1993. 193 s.

[9] GEBAUER, Lukas. *Ararat Synapse* [online]. 2009 [cit. 2011-04-24]. Unit synaser. Dostupné z WWW: <http://synapse.ararat.cz/doc/help/synaser.html>.

[10] *Lazarus Component Library* [online]. 2003 [cit. 2011-04-24]. Lazarus Documentation. Dostupné z WWW: <http://lazarus-ccr.sourceforge.net/docs/lcl/>.

[11] *Lazarus Documentation wiki* [online]. 2005 [cit. 2011-04-24]. Lazarus and Free Pascal wiki. Dostupné z WWW: <http://wiki.lazarus.freepascal.org/>.

[12] Basic Principles and Techniques for Receptor Binding. *Tocris Reviews*. 2002, 18, s. 1-8.

[13] Bylund DB, Toews ML: Radioligand binding methods: practical guide and tips. Am J Physiol Lung Cell Mol Physiol 265:L421-429.1993.

[14] Bylund DB, Deupree JD, Toews ML: Radioligand binding methods for membrane preparations and intact cells. In: Receptor Signal transduction protocols, Methods in molecular biology 259. Humana Press, New Jersey, 2004.

VII. Přílohy

Seznam příloh

Příloha č.1 Obsah přiloženého CD

Na přiloženém CD jsou kromě elektronické verze bakalářské práce umístěny následující položky:

- **data** - v této složce se nacházejí dvě měření uložená v souboru formátu programu Boreas.
- **help** - v této složce je umístěna uživatelské nápověda aplikace Boreas.
- **lazarus** - v této složce je umístěn instalační soubor IDE Lazarus ve verzi, která byla použita pro tvorbu aplikace Boreas.
- **source** - v této složce se nacházejí zdrojové kódy aplikace Boreas a také zdrojové kódy komponenty SpdoSerial.
- **install.exe** - tento soubor spustí instalační proces aplikace Boreas.

Příloha č.2 Kapitola 15 z manuálu ke scintilačnímu počítači Beckman LS 1801

CHAPTER FIFTEEN — RS232 DATA OUTPUT

15.1 INTRODUCTION

Your instrument is equipped with a standard RS232 data output port, by means of which data from the system can be fed directly to an external device such as a data logger, data buffer, or terminal computer. This Chapter contains the technical information you will need to make use of this capability.

15.2 DATA SENT

When RS-232 output is selected in an Auto Count or Special Program, all data sent to the printer is also sent to the RS-232 port. This is true for data items; headings and explanatory text are not transmitted. Also, data sent to the RS-232 port includes only those items sent to the printer; thus the selection of Print Format (Short, Standard, or Select) also affects data to the external device.

Data items appear in the RS-232 output at the same time and in the same order as they are printed out on the printer.

Some special care is recommended when using Select Format. The same cautions that apply when setting up a Select Format also apply to Select Format RS-232 data. Remember in particular that items not selected will not be printed or transmitted. Note also that in the event of a counting error or a mistake in editing, it is possible to have a run in which nothing is sent. In addition, be

aware that processing of Select Format data will be more difficult because the format of transmitted data may vary (see below).

15.3 RS-232 OUTPUT FORMAT

For easier identification, RS-232 data are grouped into blocks, with each block identified by a Block ID number, as listed in the following section.

For Standard and Short Formats, blocks having the same Block ID will always contain the same number of lines and the same data items. Zero values are generated to take the place of data items not selected or computed for a particular sample. For example, a zero is generated for the Average H# in blocks 10D and 14D if H-number was not selected or could not be calculated.

When using Select Format, most data will be sent in Select Format Blocks (Block ID = 000). Each Select Format Block contains only the data printed in the last one or two lines on the printer. The number of lines and the number of data items will <u>not</u> be the same in all blocks with ID = 000, since zero values are not generated to replace missing items.

The format for a single block is

(STX) (BLOCK ID), (CHECKSUM) (ETX) (CR) (LF)

(STX) (DATA, (DATA), ... , (DATA), (CHECKSUM) (ETX) (CR) LF)

.

.

(STX) (DATA), (DATA), ... , (DATA), (CHECKSUM) (ETX) (CR) (LF)

The following guidelines apply to the output data:

1) All characters are ASCII upper-case.

2) (STX) = 02 hex (Control - B)

3) (ETX) = 03 hex (Control - C)

4) (BLOCK ID) = (NUMBER) (NUMBER) (NUMBER)

5) (DATA) = characters:

0-9, A-Z, ! " # $ % & ' ( )

* + - . / : ; = ?

@ [ ] — (20 to 5F hex).

6) (CR) = OO hex

7) (LF) = OA hex

8) (CHECKSUM) = (ALPHANUMERIC) (ALPHANUMERIC)

where (ALPHANUMERIC) = (A-F, 0-9).

The checksum gives the two lowest hexidecimal digits of the sum of the ASCII values of the preceeding characters in the line. This includes commas but does not include the initial (STX).

As an example, consider the Block ID line for block 200:

(STX) 200, BE (ETX) (CR) (LF).

Here, ASCII (2) = 32, ASCII (0) = 30, and ASCII (,)

= 2C,

so 32 + 30 + 50 + 2C = BE, all in hexidecimal.

9) If X-OFF is chose in select format, then each line will end with:

(EXT) (X-OFF) (CR) (LF).

Here (X-OFF) = (DC3) = 13 hex.

10) A given data item will appear exactly as it is printed on the printer. Notice that some data items, such as the "ID" in Block 200, will be text strings, and some data items, such as "half-life" in Block 205, may be either an alphabetic character or a number. (The "half-life" item can be either a number, such as "10.035," or the letter "N," indicating no half-life correction.)

Some blocks consist of nothing but the Block ID line. These are used to notify the computer of errors or to identify types of data.

Note in particular Block ID 999. This block is sent during Auto Count at the end of the data for each user, when a Halt rack or new User Number card is encountered. It is <u>not</u> sent when the run is halted with a software Reset.

15.4 TECHNICAL SPECIFICATIONS

Data from the RS-232 port is sent out in standard ASCII form, upper-case only, at 300 Baud. The instrument transmits:

1 start bit

8 data bits (with no parity bit)

2 stop bits.

A device connected to the port must be set up to use:

300 Baud

1 start bit and 8 data bits

or

7 data bits + parity bit (parity will be ignored)

1 or 2 stop bits (either is acceptable)

The instrument is configured as data communications equipment (DCE), which means that it behaves like a modem. A device can be connected to the instrument in either of two ways:

1) Device configured like a DTE (ie, set up to talk to a modem or another computer). Use a standard RS-232 cable.

2) Device configured like a DCE (ie, set up to talk to a terminal or printer). Use a "modem bypass" or "null modem" cable that reverses the signal lines so that two computers or two modems can talk to one another.

The instrument will set the RTS, CTS, DSR, and DTR lines true when it is powered up. If the device connected to the instrument does not need these RS-232 handshaking lines, a basic 3-line cable (transmit data, receive data, ground) may be used. If the device requires one or more of the handshaking lines, the standard 7-line or 9-line cable supplied by most computer supply houses should be used.

15.5 HANDSHAKING

The LS instrument will respond to X-ON and X-OFF characters (ASCII codes 17 and 19, decimal). When the X-OFF character is received, data transmission is halted until an X-ON character is received. If no X-ON is received within two minutes, RS-232 transmission is halted, and "Instrument Error 206" is displayed.

The instrument will not respond to the RS-232 hardware handshake lines RTS, CTS, DSR, or DTR; all of these lines are held high by the instrument.

No ACK/NAK (acknowledge/no acknowledge) or re-transmit protocol is supported.

The Baud rate, DCE/DTE configuration, and handshake line response can be changed by your Beckman Service Representative at the time of installation, if you desire.

15.6 OUTPUT IDENTIFICATION BLOCKS

The output data blocks for each type of Data Calculation program are presented on the following pages.

Příloha č.3 Výpis z výstupu sériového portu scintilačního počítače Beckman LS 1801

```
┐200,BE┘
┐ 1,           ,  1.00,SUN,01,JAN,1984,08,44,32┘
┐ 1, 1,N  ,Y,39┘
┐ 1,N,N,N,0,47┘
┐240,C2┘
┐  0, 400, 2.00,   0.00,  0.00,   0,B8┘
┐235,C6┘
┐SUN,01,JAN,1984,08,17,DE┘
┐205,C3┘
┐                    , 1,29┘
┐Q,1.00000,DPM  ,45┘
┐H,N,N         ,88┘
┐225,C5┘
┐  4.165971,-0.0020417,-0.0000161,-0.0000000001,C7┘
┐                    ,AC┘
┐0.000,1000.,35┘
┐100,BD┘
┐  1,**- 1, 1,   6372.00, 2.51,  1.00,    1.49, 95.0,0,0,0,0,41┘
┐105,C2┘
┐45.91,13880.21,EE┘
┐100,BD┘
┐  2,**- 2, 1,   5130.00, 2.79,  1.00,    3.20, 89.0,0,0,0,0,3E┘
┐105,C2┘
┐47.31,10844.08,EC┘
┐100,BD┘
┐  3,**- 3, 1,   2860.00, 3.74,  1.00,    4.91, 83.0,0,0,0,0,46┘
┐105,C2┘
┐48.69,5873.498,0B┘
┐100,BD┘
┐  4,**- 4, 1,   3186.00, 3.54,  1.00,    6.62, 82.0,0,0,0,0,47┘
┐105,C2┘
┐48.92,6512.340,F0┘
┐100,BD┘
┐  5,**- 5, 1,   4413.00, 3.01,  1.00,    8.33, 87.0,0,0,0,0,40┘
┐105,C2┘
┐47.77,9237.819,04┘
┐100,BD┘
┐  6,**- 6, 1,   4797.00, 2.89,  1.00,   10.03, 82.0,0,0,0,0,61┘
┐105,C2┘
┐48.92,9805.304,F8┘
┐100,BD┘
┐  7,**- 7, 1,   2991.00, 3.66,  1.00,   11.74, 91.0,0,0,0,0,62┘
┐105,C2┘
┐46.84,6385.308,FB┘
┐100,BD┘
┐  8,**- 8, 1,   2696.00, 3.85,  1.00,   13.45, 79.0,0,0,0,0,6D┘
┐105,C2┘
┐49.61,5434.642,F4┘
┐100,BD┘
┐  9,**- 9, 1,   5885.00, 2.61,  1.00,   15.16, 82.0,0,0,0,0,65┘
┐105,C2┘
┐48.92,12029.23,EE┘
┐100,BD┘
┐ 10,**-10, 1,   5909.00, 2.60,  1.00,   16.86, 87.0,0,0,0,0,7E┘
┐105,C2┘
┐47.77,12369.43,F9┘
┐100,BD┘
```

```
┐ 11,**-11, 1,   4224.00, 3.08,  1.00,   18.57, 84.0,0,0,0,0,75┘
┐105,C2┘
┐48.46,8715.904,FC┘
┐100,BD┘
┐ 12,**-12, 1,   4538.00, 2.97,  1.00,   20.27, 82.0,0,0,0,0,7A┘
┐105,C2┘
┐48.92,9275.895,08┘
┐999,D7┘
```

Příloha č.4 Výpis z tiskárny připojené na scintilační počítač Beckman LS 1801


```
USER: 1 ID:3H5MIN        PRESET TIME:  5.00            FRI 21 JAN 2011 12:35
SAMPLE REPEAT: 1 CYCLE REPEAT: 1 SCR:N   RS232:N
H#: 1 AQC:N QCF:N RCM:N
CHANNEL 1-LL:  0 UL: 400 2SIGMA: 2.00 BKG SUB:  0.00 BKG 2SIG:  0.00 LSR:

SINGLE LABEL DPM, SET UP ON SUN 01 JAN 1984 08:05
UNKNOWN ID:                          UNKNOWN REPLICATES: 1
UNKNOWN NORM FACTOR:@ 1.00000        UNKNOWN UNITS:DPM
QUENCH MODE:H  CALCULATE COEFF:N  HALF LIFE(DAYS):N
QUENCH COEFF A:  4.165971  B:-0.0020417  C:-0.0000161  D:-0.0000000061
STANDARD ID:
QUENCH LIMITS LOW:0.000  HIGH:1000.

SAM  POS  CH     CPM     2SIG%  TIME   EL TIME AVG H#                    EI
 1 **-1  1   12607.30   1.97   0.80     1.29 106.0
                                %EFF:43.01 ISO1 DPM  :29314.41
 2 **- 2  1   13208.75   1.95   0.80     2.75 105.0
                                %EFF:43.25 ISO1 DPM  :30539.43
 3 **- 3  1   13854.67   1.96   0.75     4.09 104.0
                                %EFF:43.49 ISO1 DPM  :31053.70
 4 **- 4  1   14987.14   1.95   0.70     5.44 104.0
                                %EFF:43.49 ISO1 DPM  :34457.50
 5 **- 5  1   10410.00   1.96   1.00     7.10 105.0
                                %EFF:43.25 ISO1 DPM  :24068.55
 6 **- 6  1   10273.00   1.97   1.00     8.76 105.0
                                %EFF:43.25 ISO1 DPM  :23751.80
 7 **- 7  1   11611.11   1.96   0.90    10.30 105.0
                                %EFF:43.25 ISO1 DPM  :26845.59
 8 **- 8  1   13504.00   1.99   0.75    11.69 105.0
                                %EFF:43.25 ISO1 DPM  :31222.07
 9 **- 9  1   11064.21   1.95   0.95    13.29 107.0
                                %EFF:42.76 ISO1 DPM  :25872.55
10 **-10  1   11230.00   1.99   0.90    14.84 105.0
                                %EFF:43.25 ISO1 DPM  :25964.44
11 **-11  1   12134.12   1.97   0.85    16.34 106.0
                                %EFF:43.01 ISO1 DPM  :28213.72
12 **-12  1   13357.33   2.00   0.75    17.74 104.0
                                %EFF:43.49 ISO1 DPM  :30710.34
13 **- 1  1    5059.00   1.99   2.00    20.50 108.0
                                %EFF:42.52 ISO1 DPM  :11897.72
14 **- 2  1    5907.06   2.00   1.70    22.89 106.0
                                %EFF:43.01 ISO1 DPM  :13734.83
15 **- 3  1    5646.11   1.98   1.80    25.37 104.0
                                %EFF:43.49 ISO1 DPM  :12981.18
16 **- 4  1    6641.29   1.97   1.55    27.59 106.0
                                %EFF:43.01 ISO1 DPM  :15442.04
17 **- 5  1    5111.00   1.98   2.00    30.22 102.0
                                %EFF:43.98 ISO1 DPM  :11621.11
18 **- 6  1    4904.86   1.99   2.05    32.97 103.0
                                %EFF:43.74 ISO1 DPM  :11214.30
19 **- 7  1    5667.78   1.98   1.80    35.40 105.0
                                %EFF:43.25 ISO1 DPM  :13104.25
20 **- 8  1    4647.73   1.98   2.20    38.29 108.0
                                %EFF:42.52 ISO1 DPM  :10930.47
```

Příloha č.5 Zdrojový kód metody TData.VypoctiTabulku, která se stará o výpočet dat pro export

```
procedure TData.VytvorTabulku(var Grid: TStringGrid);
var i,j: word;
    X,L: real;
begin
Grid.RowCount:=1;
Grid.ColCount:=1;
Grid.FixedCols:=1;
Grid.FixedRows:=1;

Grid.RowCount:=SAM+1;

case Method of
mOpa : begin
       Grid.ColCount:=5;
       Grid.Rows[0].CommaText:='#,Bmax[fmol],"Bmax[fmol/mg
prot.]","Bílkovina       [mol/mg]",Skupina';

       x:=0;
       for i:=0 to ssam-1 do x:=x + avg(NSAM+(i*SSD),SSD);
       L:=x/ssam;
       L:=L/ASS/VOL;

       for i:=0 to sam-1 do
         begin
           X:=avg((i*SD*2),SD)-avg((i*SD*2)+SD,SD);
           Grid.Cells[0,i+1]:=IntToStr(i+1);
           Grid.Cells[1,i+1]:=ToStr((X/ASS)*(L+KD)/L);
           Grid.Cells[2,i+1]:=ToStr(((X/ASS)*(L+KD)/L)/StrToFloat(Ref[
i]));
           Grid.Cells[3,i+1]:=ToStr(Ref[i]);
           Grid.Cells[4,i+1]:=IntToStr(FindGroup(i+1)+1);
         end;
       end;

mSat: begin
      Grid.ColCount:=(SD*2)+2;
      Grid.Rows[0].CommaText:='#,[L]';
      for i:=1 to SD*2 do Grid.Cells[1+i,0]:=IntToStr(i);

      for i:=0 to SAM-1 do
        begin
          Grid.Cells[0,i+1]:=IntToStr(i+1);
          Grid.Cells[1,i+1]:=ToStr((avg(NSAM+(i*SSD),SSD) -
                     avg(2*SD*i,SD))/CONS);

          for j:=0 to (SD*2-1) do
      Grid.Cells[2+j,i+1]:=ToStr(StrToFloat(Dpm[i*(SD*2)+j])/CONS);
        end;
      end;
   end;
Grid.AutoSizeColumns;
end;
```

Příloha č.6 Zdrojový kód metody TData.VytvorGraf, která se stará o vykreslení grafu

```
procedure TData.VytvorGraf(var Chart: TChart;var FBar: TBarSeries;var
FLine: TLineSeries);
var X,Y,L: real;
    i,j,s,e: word;
begin
FBar.Clear;
FLine.Clear;

case Method of
mOpa : begin
       Chart.BottomAxis.Title.Caption:='Skupiny';
       Chart.LeftAxis.Title.Caption:='Bmax [fmol/mg. prot.]';
       Chart.BottomAxis.Visible:=False;
       Chart.LeftAxis.Title.Visible:=True;

       x:=0;
       for i:=0 to ssam-1 do x:=x + avg(NSAM+(i*SSD),SSD);
       L:=x/ssam;
       L:=L/ASS/VOL;

       for i:=0 to Group.Count-1 do
       begin
       s:=StrToInt(Group[i])-1;
       if (i=Group.Count-1) then e:=sam-1 else
e:=StrToInt(Group[i+1])-1;

       Y:=0;

       for j:=s to e do
         begin
           X:=avg((j*SD*2),SD)-avg((j*2*SD)+SD,SD);
           Y:=Y+(((X/ASS)*(L+KD)/L)/StrToFloat(Ref[j]));
         end;

       Y:=Y/(e-s+1);

       FBar.AddXY(i+1, Y, ToStr(Y), SerColors[i mod
Length(SerColors)]);
       end;
  end;
mSat : begin
       Chart.BottomAxis.Title.Caption:='F [nmol]';
       Chart.LeftAxis.Title.Caption:='B [fmol/mg. prot.]';
       Chart.BottomAxis.Title.Visible:=True;
       Chart.BottomAxis.Visible:=True;
       Chart.LeftAxis.Title.Visible:=True;

       for i:=0 to SAM-1 do
         begin
           x:=avg(NSAM+(i*2),2) - avg(2*SD*i,SD);
           y:=(avg(i*(2*SD),SD) - avg(i*(2*SD)+SD,SD))/CONS;
           FLine.AddXY(X, Y, ToStr(Y), clRed);
         end;

       end;
    end;
end;
```